KEMI-TORNION AMMATTIKORKEAKOULU

Vaadin

Suomalainen Java-pohjainen web-ohjelmistokehys

Aalto Pekka

Tietotekniikan koulutusohjelman opinnäytetyö
Ohjelmistotekniikka
Insinööri(AMK)

KEMI 2011

ALKUSANAT

Kiitokset opinnäytetyön aiheen valinnasta Kemi-Tornion ammattikorkeakoulun Marko Heikkilälle ja Antti Niemelälle. Haluan kiittää myös ohjaaja Tapani Ruokasta ja koulutusvastaava Soili Mäkimurto-Koivumaata joustavasta suhtautumisesta opintojeni viimeisten vaiheiden loppuun saattamisessa.

TIIVISTELMÄ

Kemi-Tornion ammattikorkeakoulu, Tekniikan ala

Koulutusohjelma	Tietotekniikka
Opinnäytetyön tekijä	Pekka Aalto
Opinnäytetyön nimi	Vaadin - Suomalainen Java-pohjainen web-ohjelmistokehys
Työn laji	Opinnäytetyö
päiväys	30.11.2011
sivumäärä	45 + 3 liitesivua
Opinnäytetyön ohjaaja	DI Tapani Ruokanen, yliopettaja
Yritys	Kemi-Tornion ammattikorkeakoulu
Yrityksen yhteyshenkilö/valvoja	YTM Soili Mäkimurto-Koivumaa, koulutuspäällikkö

Opinnäytetyön lähtökohtana oli tutkia suomalaista web-ohjelmistokehystä ja testata käytännössä sen ominaisuuksia ja toimintoja. Ennen varsinaista asennusta ja testaamista työssä vertailtiin lähdetietojen perusteella vastaavien ohjelmistokehysten ominaisuuksia ja etsittiin perusteita Java-kielen käytölle. Lisäksi tutkittiin ohjelmistokehyksen ja sillä tehtävän sovelluksen arkkitehtuuria.

Testauksessa asennettiin tietokoneelle tarvittava ohjelmistopaketti kehitysympäristöineen ja luotiin Book of Vaadin -ohjekirjan mukainen testiprojekti. Kyseinen kirja oli pääasiallinen tiedon lähde tässä opinnäytetyössä, vaikka hyvin paljon muuta teoriatietoa tutkittiinkin monista eri internetlähteistä. Opinnäytetyössä ei tutkittu muita web-ohjelmistokehyksiä käytännön tasolla vaan pelkästään teoriassa ja vertailevien internet-lähteiden perusteella. Käyttökokemuksia saatiin siis vain Vaadin-ohjelmistokehyksestä ja sen toiminnoista.

Tutkimustyön pääasiallisena tuloksena voidaan pitää sitä saatua hyödyllistä tietoa, mitä tämäntyyppisellä Java-pohjaisella ohjelmistokehyksellä on mahdollista tehdä. Käytännön kokemusta saatiin riittävästi ohjelmiston toiminnasta, joskaan ei vielä ammattimaisen kehittäjän tasolla. Aloittelijan johtopäätöksenä voidaankin todeta, että vaikka Vaadin oli suhteellisen helppokäyttöinen, se vaatii selvästi enemmän ohjelmointitaitoja ja myös laajempaa ohjelmisto-osaamista kuin esimerkiksi suositut avoimen lähdekoodin julkaisujärjestelmät. Näiden taustojen valossa asetetut tavoitteet saavutettiin kohtalaisen hyvin.

Asiasanat: Java, avoin lähdekoodi, sovelluskehykset, verkko-ohjelmointi, Ajax-ohjelmointi.

ABSTRACT

Kemi-Tornio University of Applied Sciences, Technology

Degree Programme	Information Technology
Name	Pekka Aalto
Title	Vaadin - Finnish Java based Web Application Framework
Type of Study	Bachelorôs Thesis
Date	30 November 2011
Pages	45 + 3 appendixes
Instructor	Tapani Ruokanen, M.Sc. (Tech.), Principal Lectrurer
Company	Kemi-Tornio University of Applied Sciences
ContactPerson/Supervisor from Company	Soili Mäkimurto-Koivumaa, M.Sc. (Econ.), Education Manager

The objective of this study was to examine Finnish web application framework and test its features and functionalities in practice. Before actual installation and testing, comparable application frameworksô features were compared and basis for usage of Java programming language was researched. In addition, the architecture of Vaadin framework was examined as well as the architecture of an application made with Vaadin.

During test period, all necessary integrated applications with the development environment were installed and a test project was created according to the Book of Vaadin ï manual. The book was main information source in this study, although many internet sources were explored to get valid information in theory. Other web application frameworks were not tested in practice. They were just examined in theory and based on the comparison of some internet sources. Experience in practice was received only from Vaadin framework and its functions.

As a result of this research, a lot of useful information of possibilities to utilize this kind of Java based framework was gathered. Also experience of functionalities in practice was gained sufficiently. Without having skills of professional level, it appears that even though Vaadin framework was relatively easy to use, it requires more programming skills and also advanced integrated applications skills than e.g. popular open source content managing systems. Considering these backgrounds, the objectives were achieved fairly well.

Keywords: java, open source, application frameworks, web-programming, ajax-programming.

SISÄLLYSLUETTELO

KÄYTETYT MERKIT JA LYHENTEET

ActionScript	Macromedian kehittämä oliopohjainen kieli, joka pohjautuu samaan standardiin ja kokoelmaan kuin JavaScript.
AJAX	*Asynchronous JavaScript And XML*. Ryhmä web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia ja ns. rikkaampia sovelluksia (RIA).
Apache TomCat	Java-palvelin ja Servlet-säiliö, jolla toteutetaan HTTP-palvelin ympäristössä Java-kielisten ohjelmien ajo.
API	*Application Programming Interface*. Ohjelmointirajapinta, joka määrittää ohjelmien/sovellusten keskinäisen vuorovaikutuksen.
BSD (lisence)	*Berkeley Software Distribution*. Eräs avoimen lähdekoodin vapaa ohjelmistolisenssi.
CMS	*Content Management System*. Sisällönhallintajärjestelmä, usein myös www-julkaisujärjestelmä, joka koostuu verkkopalvelukokonaisuuteen integroiduista ohjelmistoista ja sovelluksista.
CSS	*Cascading Style Sheets*. Yksinkertainen tyylikieli, jonka avulla määritellään ja muotoillaan WWW -dokumenttien ulkoasu.
DB	*Database*. Tietokanta.
Devoxx	*The Java Community Conference* (Vuosittainen konferenssi).
DOM	*Document Object Model*. Alustariippumaton ohjelmointirajapinta, joka mahdollistaa kommunikoinnin (X)HTML ja XML dokumenttien kanssa ohjelmoinnissa määritellyn objektin avulla.
EJB	*Enterprise JavaBeans.* Palvelinpuolen komponenttiarkkitehtuuri ja malli, jossa business-logiikka erotetaan omaksi toiminnalliseksi kokonaisuudeksi.
GUI	*Graphical User Interface*. Graafinen käyttöliittymä, joka koostuu graafisista elementeistä kuten tekstistä, ikkunoista, kuvakkeista, valikoista jne.
GWT	*Google Web Toolkit*. Avoimen lähdekoodin kokoelma työkaluja, joilla web-kehittäjät voivat luoda ja ylläpitää monimutkaisia Java-sovelluksia.
(X)HTML	*(eXtensible) Hypertext Markup Language*. Kuvauskieli, joka määrittelee miten informaatio näytetään web-selaimessa.
HTTP(S)	*Hypertext Transfer Protocol (Secure)*. Protokolla, joka säätelee kaikkea informaation siirtoa Internetissä HTTP-palvelinten ja web-selainten välillä. Secure = Suojattu siirto.
i18n & L10n	*internationalization and Localization*. Lyhennetyt termit tarkoittavat ohjelmistojen sopeuttamista eri maantieteellisiin ympäristöihin kielellisesti, kulttuurillisesti ja teknillisesti alueelliset kehityserot huomioiden.
IDE	*Integrated Development Environment*. Integroitu kehitysympäristö, eli ohjelma tai joukko ohjelmia, jolla ohjelmoija toteuttaa ohjelmistoa.
J2EE	*Java 2 Enterprise Edition.* Erityisesti palvelimille tarkoitettu Java-alustan (kehittäjä)versio.

jar	*Java Archive*. Java-arkistotiedostoformaatti.
javadoc	Java-generaattori, jolla tuotetaan Java-kielisestä lähdekoodista ohjelmointirajapinnan dokumentaatiota HTML-formaatissa.
JavaScript	Web-ympäristössä käytettävä komentosarjakieli, jolla lisätään web-sivustoihin dynaamista toiminnallisuutta.
jdb	*Java Debugger*. Javan testaustyökalu.
JDK	*Java Development Kit*. Java-kehittäjille tarkoitetut työkalut sovellusten tekemiseen.
JNDI	*Java Naming and Directory Interface*. Javan nimeämis- ja hakemistopalveluiden rajapinta.
JRE	*Java Runtime Environment*. Javan ajoympäristö, joka tarvitaan Java-sovellusten suorittamiseen ja ajamiseen.
JSP	*JavaServer Pages*. Javaan perustuva menetelmä luoda HTML- ja XML-muotoisia web-sivuja.
JSON	*JavaScript Object Notation*. Tiedonsiirtomuoto, jota käytetään pääosin palvelimen ja web-sovelluksen välillä. Voidaan käyttää myös muilla ohjelmointikielillä tehdyissä ohjelmissa kuin pelkästään JavaScript-ohjelmissa.
JVM	*Java Virtual Machine*. Virtuaalikone (komponentti), jonka avulla voidaan suorittaa Javan tavukoodia ja ajaa käännettyjä Java-ohjelmia.
MVC	*Model-View-Controller*. Ohjelmistoarkkitehtuurityyli, jossa graafinen käyttöliittymä erotetaan sovellusaluetiedosta.
MXML	Macromedian kehittämä XML-pohjainen käyttöliittymän merkintäkieli, jota käytetään yhdessä Actionscripitin kanssa.
ORM	*Object-relational mapping*. Ohjelmointitekniikka, jolla muunnetaan yhteensopimaton data olio-ohjelmointiin sopivaksi.
Push/Pull (MVC)	MVC-arkkitehtuurityylien mukaiset vastaus-pyyntö-toiminnot, jotka on jaettu kommunikoinnissa ns. työntö- ja vetomalliin.
REST	*Representational State Transfer*. Arkkitehtuurityyli, jolla pyritään varmistamaan yhteentoimivuus sellaisissa hajautetuissa järjestelmissä, joissa eri osapuolet kehittyvät ja muuttuvat itsenäisesti toisistaan riippumatta.
RIA	*Rich Internet Application*. Internet-sovellus, jonka toteutuksessa on käytetty korkeaa interaktiotasoa tarjoavia komponentteja, ja muistuttaa täten perinteistä työpöytäsovellusta.
RWX	*The Rich Web Experience* (Vuosittainen konferenssi).
SDK	*Software Development Kit*. Kokelma kehitystyökaluja sovellusten tekemiseen tiettyä ohjelmaa tai ohjelmistokehystä varten.
TSSJS	*TheServerSide Java Symposium* (Vuosittainen konferenssi).
UIDL	*User Interface Definition (Description) Language*. Java-pohjainen kieli monimutkaisten ja korkean abstraktiotason käyttöliittymien sisällön sekä niiden muutosten esittämiseen.
URI	*Unified Resource Identifier*. Resurssin nimi ja/tai sijainti.
URL	*Unified Resource Locator*. Nimeämiskäytäntö määriteltäessä Internet-avaruutta. Määrittää lähteen osoitteen.

WAR	*Web application Archive*. JAR-tiedosto, joka luokittelee mm. kokoelman Java-servlettejä, -luokkia, XML-tiedostoja web-sovellusta varten.
www	*World Wide Web*. Maailmanlaajuinen verkossa toimiva hajautettu järjestelmä, joka sisältää tietolähteitä ja resursseja.
WYSIWYG	*What You See Is What You Get*. Käsite kuvaa hyvin usein tekstinkäsittelyohjelmia, HTML- ja muita vastaavia editoreja. Editorilla työskentely ei edellytä varsinaisen muotoilukoodin tuntemista, vaan koodi generoituu muokkauksen yhteydessä.
XUL(Runner)	Kokeellinen teknologia Mozillan kehittämälle ajoympäristölle, jolla ajetaan Mozillan XUL-pohjaisia, XML-merkintäkielisiä käyttöliittymän sovelluksia.

1. JOHDANTO

Ohjelmistotekniikoiden kehittyminen on johtanut siihen, että web-sovelluksissa voidaan käyttää useita eri ohjelmointikieliä ja toteutustapoja tarpeen mukaan. Erilaisten teknologioiden ja arkkitehtuurityylien yhteensovittaminen sekä palveluiden kasvava vaatimustaso tuovat omat haasteensa hajautetuille järjestelmille, ohjelmistoille ja niiden rajapinnoille.

Tämän opinnäytetyön lähtökohtana oli tutustua, tutkia ja testata suomalaista Vaadin-ohjelmistokehystä sekä tarkastella sen ratkaisuja osana dynaamisia web-palveluita. Vertailua muihin eri Java-pohjaisiin ohjelmistokehyksiin ei tehty toiminnan ja käytännön tasolla, vaan ainoastaan teorian tasolla ja käytettävissä olevien lähdetietojen perusteella.

Alustavan teoriaosuuden ja osin liitteidenkin tarkoitus on syventää aiheeseen liittyvää yleistietoa ja toivottavasti myös herättää lukijassa mielenkiintoa ja ehkä myös kysymyksiä liittyen Vaadin-ohjelmistokehykseen tehtyihin ratkaisuihin. Työn tavoitteena on käyttökokemuksen ja tiedon saamisen lisäksi muodostaa kokonaisnäkemys yhden Java-pohjaisen ohjelmistokehyksen tavasta toimia.

Avoimen lähdekoodin ohjelmistona Vaadin on vapaasti ladattavissa, kuten myös muut siihen liitetyt tarvittavat ohjelmat ja kehitysympäristöt. Ohjelmistot voi asentaa mihin tahansa yleiseen käyttöjärjestelmään, joten laitteistovaatimukset ovat minimaaliset; toimiva tietokone ja verkkoyhteys riittävät.

Ennen varsinaista ohjelmiston asennusta ja testausta syvennytään arkkitehtuureihin ja niihin rakenteellisiin elementteihin, joita Vaadin käyttää Java-alustan päällä toimiakseen. Samalla tarkastellaan ohjelmistokehyksellä ajettavan sovelluksen arkkitehtuuria ja miten sen teoriassa tulisi toimia komponenttien avulla.

Ensimmäisessä käytännön osuudessa esitellään ja asennetaan kaikki tarvittavat ohjelmistot. Sen jälkeen luodaan ohjekirjan mukaan ensimmäinen testiprojekti (kevyt sovellus) vaihe vaiheelta. Tällä varmistetaan ohjelmointikehyksen kokonaisvaltainen toiminta muiden

integroitujen ohjelmien kanssa. Sen jälkeen testataan vielä graafista käyttöliittymää ja yksinkertaisen sovelluksen tekemistä WYSIWYG-editorilla.

Tiedon etsimiseen on käytetty runsaasti internetlähteitä, myös sellaisia, joita ei ole käytetty varsinaisesti apuna tekstin tuottamiseen, vaan alustavan teoriatiedon vertailuun ja yleensä aiheeseen liittyvän tiedon ymmärtämiseen. Pääasiallisena tietolähteenä on käytetty englanninkielistä Book of Vaadin -ohjekirjaa.

2. AVOIMEN LÄHDEKOODIN OHJELMISTOKEHYKSET

Asiakas/palvelin -arkkitehtuuri (client/server) muodostuu palvelinprosesseista ja näille yhteisestä yhteyskäytännöstä eli protokollasta. Prosessit voivat olla ohjelmia tai niiden osia. Palvelin arkkitehtuuri on kehittynyt myös ns. monikerrosmalliksi, jossa on mukana yleensä tietokantapalvelin dynaamisten sisältöjen esittämistä varten. Tämä tarkoittaa sitä, että varsinainen ohjelmistokehys, jolla työskennellään, on usein monen ohjelman tai sovelluksen integroitu rypäs. Usein puhutaan samassa yhteydessä pelkästään ohjelmistokehyksestä, joka vaatii yleensä toimiakseen myös asennettavan kehitysympäristön. Kaikki nämä toisiaan tukevat ohjelmat on usein mahdollista ladata ilmaiseksi, koska niiden kehitystyö pohjautuu avoimeen lähdekoodiin ja sitä kautta käyttövapaisiin lisensseihin. /1/

Avoimen lähdekoodin ohjelmistokehyksiä on tarjolla tänä päivänä monenlaisia, riippuen mitä ohjelmointikieltä halutaan käyttää ja mihin tarkoitukseen esimerkiksi web-sovelluksia tai -palveluita halutaan tehdä. Ohjelmistosta riippuen, toimintojen ohjausta ja käynnistystä ts. kontrollia, voidaan painottaa eri tavalla asiakkaan ja palvelimen välillä.

Asiakas/palvelin -arkkitehtuuri on yksinkertainen malli kuvata hajautettua ohjelmistoa. Javan ohjelmointirajapinta (API) tarjoaa yhden helpon ja nopean tavan toteuttaa asiakas/palvelin-ratkaisuja Java-kielellä, koska se sisältää Javan valmiit kirjastot ja kommunikoi eri ohjelmistotasojen välillä. /1/

2.1. Java vertailussa

HTML-kielen ja www:n julkistuksen myötä Sun Microsystems alkoi kehittää Oak-kielen pohjalta Javaa 1990-luvun loppupuolella. Java on ohjelmointikieli, joka on pyritty tekemään laitteistoriippumattomaksi graafisten ohjelmien tekovälineeksi. Java onkin nykypäivänä tullut niin suosituksi kieleksi, että lähes jokaisessa uudessa selaimessa on Java-tulkki ja yhä useammilla www-sivuilla on käytetty jotakin Javalla tehtyä sovellusta. /12/

Ohjelmointikielien suosiosta on lukuisia ristiriitaisia tuloksia mittaus- ja arviointitavasta riippuen, koska ohjelmistokehittäjien tarkkaa lukumäärää on vaikea arvioida. TIOBE Softwaren ylläpitämässä indeksissä Java on edelleen suosituin kieli, vaikka viimeaikaisista uudistuksista huolimatta Java-ohjelmointikieli näyttää menettävän suosiotaan. /6/

òTIOBE:n uusimman (2011) ohjelmointi-indeksin mukaan Javan suosio laski syyskuussa 17,9 prosenttiin elokuun 18,8 prosentista. Se säilytti yhä paikkansa suosituimpana ohjelmointikielenä, mutta C-ohjelmointikieli peittoaa sen seuraavassa kuussa, jos suunta jatkuu laskevana, arvioi TIOBE. Nyt toiseksi suosituimman C:n osuus oli 17,7 %.ò/6/

òJava on pitänyt suosituimpien ohjelmointikielien kärkipaikkaa vuodesta 2001 lähtien, vuosien 2004, 2005 ja 2010 lyhyehköjä taukoja lukuun ottamatta. TIOBE:n mukaan Javan suosion lasku johtuu siitä, että se kehittyy liian hitaasti muihin ohjelmointikieliin verrattuna. Ohjelmointikielten kärkikastissa ovat Javan lisäksi C++, PHP, C#, Objective-C, Visual Basic, Python, Perl ja JavaScript.ò /6/

2.1.1. Javan edut ja haitat

Mittaustuloksista huolimatta kaikilla ohjelmointikielillä on hyvät ja huonot puolensa. Kehityskilpailu on kovaa ja tänä päivänä on paljon vaihtoehtoisia alustoja ja ohjelmistokehyksiä erilaisille järjestelmille. Pitää siis osata valita tarkoitukseen sopiva sovellus- tai ohjelmistopaketti, myös omat ohjelmointitaidot huomioiden.

Lukuisten internetlähteiden pohjalta voidaan vetää johtopäätöksiä Javan eduista ja haitoista. Etuja näyttäisi olevan huomattavasti enemmän ja käyttäjäkokemukset yhteneväiset. Java-kielen yleisyyden lisäksi Java sopii hyvin myös aloittelijalle. Selviä etuja ovat muun muassa seuraavat alla luetellut seikat.

Java on yksinkertainen ohjelmointikieli

Java on suunniteltu helposti käytettäväksi ja opittavaksi kieleksi. Ohjelmoijan ei tarvitse huolehtia muistin varauksesta ja vapauttamisesta, vaan se hoidetaan automaattisesti.
Javassa on myös automaattinen roskienkeruujärjestelmä. /3/

Java on oliopohjainen

Olio-ohjelmointi on nykyisin suosituin ohjelmointiparadigma, jossa lähestymistapa ohjelmointiin on oliot ja niiden väliset suhteet. Tästä syystä Java-kieli on modulaarinen ja koodi helposti käytettävissä uudelleen. /3/

Java on alusta- ja laitteistoriippumaton

Java-kielisen ohjelman voi ajaa monissa eri järjestelmissä ilman yhteensopivuusongelmia. Varsinkin web-sovelluksissa tällainen laajennettavuus ja joustavuus ovat erittäin tärkeitä ominaisuuksia, kun puhutaan hajautetuista järjestelmistä. Tässä tapauksessa alustariippumattomuudella tarkoitetaan, että kirjoitettu ohjelmakoodi käännetään ns. tavukoodiksi (bytecode), joka on tietynlainen väliversio ohjelmasta. Tätä tavukoodista versiota voidaan ajaa järjestelmälle ja laitealustalle sopivalla Java-virtuaalikoneella (JVM) eli tulkilla, joka tulkkaa käännöksen koneen ymmärtämään muotoon. Itse kääntäminen tavukoodiksi tapahtuu siis vain kerran, mutta tulkkaus tapahtuu joka kerta, kun ohjelma ajetaan. Jokainen Java-lähdekoodi voidaan kääntää millä tahansa koneella, johon on asennettu Java-kääntäjä. Käännetyt tavukoodit voidaan suorittaa jokaisessa koneessa, jossa on Java-tulkki asennettuna. /3/

Java on turvallinen

Javan suunnittelussa turvallisuus on eräs avaintekijä, joka on huomioitu mahdollisten virheiden tarkistuksina jo aikaisessa vaiheessa. Java-kääntäjät pystyvät havaitsemaan ongelmat tehokkaasti ennen ohjelman ajoa. Palvelinpuolen ratkaisut tukevat myös turvallisuutta ja luotettavuutta, vielä kun Apache Tomcat ï palvelin on liitetty osaksi laajempaa Apache-palvelinohjelmistoa sitä varten kehitetyn moduulin avulla. /3/

Java on monisäikeinen

Java pystyy suorittamaan ohjelman sisällä monia tehtäviä samanaikaisesti. Tämä on erityisen hyvä piirre dynaamisissa web-sovelluksissa ja muissa monitasoisissa ratkaisuissa, joissa luodaan lisätoimintoja erilaisten komponenttien avulla. /3/

Javan haittoja ohjelmointikielenä on luetteloitu yllättävän vähän. Seuraavaksi on lueteltu niistä muutamia.

Javan suorituskyky

Java-kieli voi olla selvästi hitaampi ratkaisu verrattuna vastaavan ohjelman tekemiseen ja kääntämiseen esimerkiksi C tai C++ kielellä. Tehokkuuden laskennallinen ero johtuu usein siitä, että Java-tulkin käyttö ja kääntämisen välivaihe vaatii prosessilta ylimääräistä aikaa. Toisaalta muistin käyttö ja resurssien varaus vaikuttaa myös oleellisesti nopeuteen. /3/

Javan ulkoasu

Java-kieliset GUI-sovellukset voivat poiketa hyvin paljon ulkoasultaan ns. natiiveista ohjelmista. Näkymä voi poiketa esim. eri selaimissa tai mitä alun perin on toivottu, vaikka koodi olisikin moitteetonta. Tämä voi johtua monista tekijöistä, mm. selaimesta, JVM-komponentista ja Java-palvelimesta, joilla ohjelmia ajetaan. /3/

Java voi olla työläs

Siinä, missä Javan etuihin luetaan koodin yksinkertainen ja helposti ymmärrettävä syntaksi, voi koodaus tuottaa pienissä ohjelmissa turhaa työtä. Ohjelmien koot kasvavat helposti suuriksi. Lisäksi ohjelmoijalla on vähemmän mahdollisuuksia optimoida koodia. Ohjelmia ei voi enää suunnitella funktionaaliselta pohjalta, vaan ohjelmien rakenteen täytyy noudattaa yhden paradigman luokka-oliomallia. /3/

2.2. Java-pohjaiset web-ohjelmistokehykset vertailussa

Nopein ja helpoin tapa tutustua tarjolla oleviin ohjelmistokehyksiin sekä niiden ominaisuuksiin on Wikipediaan tehdystä vertailusivusta (Comparison of Web application frameworks). Sinne on muun muassa listattu erityyppisiä Javaa käyttäviä ohjelmistokehyksiä tällä kirjoitushetkellä kaikkiaan 37, joista taulukkovertailuun on otettu mukaan 23 (Liitteet 1/1-2/1). Suurin osa luetelluista tuotteista on avoimen lähdekoodin

ohjelmistoja ja perustuvat käyttöoikeuksiltaan erityyppisiin ns. vapaisiin lisensseihin. Listassa ei välttämättä ole kaikkia varteenotettavia ohjelmistokehyksiä, joita on tarjolla. Joidenkin lähteiden ja esitettyjen laskelmien mukaan erityyppisiä Java-pohjaisia ohjelmistokehyksiä olisi tänä päivänä reilut 60. Kattavia ominaisuuksien vertailuja löytyy silti melko vähän ja ne eivät ole välttämättä aina kovinkaan puolueettomia. Yksi sellainen on myös suomalaisen Vaadin-ohjelmistokehyksen sivustolta löytyvä vertailu, jossa on mukana nimenomaan sen kanssa kilpailevia, samankaltaisia Java-pohjaisia web-ohjelmistokehyksiä (Liite 2). /11/, /13/

Amerikkalainen web-arkkitehtuurien konsultti Matt Raible on kuitenkin tehnyt vertailua jo pitemmän aikaa erilaisista web-ratkaisuista ja -ohjelmistokehyksistä. Hän on kasannut esityksiinsä niitä pääpiirteitä, mitä ohjelmistoilta ja niiden suunnittelulta vaaditaan tänä päivänä ja mitä eri tekniikoiden kehitys tuo tullessaan. Kuvan 1 vertailussa hän on käyttänyt tekemäänsä taulukkoa eri ohjelmistokehyksien pisteytyksistä erilaisten kriteerien perusteella. Lisäksi kuvassa 2 sama vertailu on tehty painotuskertoimilla, joilla otetaan huomioon eri ominaisuuksien tärkeys ohjelmakehysten kehittäjille.

Vertailu ei ole yksiselitteisen luotettava, vaan enemmänkin omiin ja muiden kehittäjien kokemuksiin perustuva näkemys. Tuloksia tukee kuitenkin Java-yhteisöjen ohjelmistovalinnat vuosittaisissa kokoontumisissaan (TSSJS, RWX, Devoxx). Vertailussa on mukana myös muilla ohjelmointikielillä kuin pelkästään Javalla toimivia sovelluksia; Grails, Rails, Flex ja Lift. Yhteisenä tekijänä kaikissa vertailun kehyksissä on JVM (Java Virtual Machine), jonka òpäälläò voidaan ajaa joissain tapauksissa myös muilla ohjelmointikielillä tehtyjä ohjelmia. JVM-alustalla toimivia ohjelmointikieliä ovat esimerkiksi Groovy (Grails), JRuby (Ruby on Rails) ja Scala (Lift). Adobe Flexin ohjelmointikielet ActionScript ja MXML on tuettu Java EE -alustalla, joka on ollut integroituna ko. projektiin jo sen kehityksen alusta alkaen. /8/

JVM Web Framework Matrix

Matrix | Weighted

Criteria	Struts 2	Spring MVC	Wicket	JSF 2	Tapestry	Stripes	GWT	Grails	Rails	Flex	Vaadin	Lift	Play
Developer Productivity	0.50	0.50	0.50	0.50	1.00	0.50	1.00	1.00	1.00	0.00	1.00	0.50	1.00
Developer Perception	0.50	1.00	1.00	0.00	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Learning Curve	1.00	1.00	0.50	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00
Project Health	0.50	1.00	1.00	1.00	0.50	0.50	1.00	1.00	1.00	0.50	1.00	1.00	1.00
Developer Availability	0.50	1.00	0.50	1.00	1.00	0.50	1.00	0.50	1.00	1.00	0.50	0.00	0.50
Job Trends	1.00	1.00	0.50	1.00	0.50	0.00	1.00	0.50	1.00	1.00	0.00	0.00	0.50
Templating	1.00	1.00	1.00	0.50	1.00	1.00	0.50	1.00	1.00	0.50	0.50	0.50	0.50
Components	0.00	0.00	1.00	1.00	1.00	0.00	0.50	0.50	0.50	1.00	1.00	0.00	0.00
Ajax	0.50	1.00	0.50	0.50	0.50	0.50	1.00	0.50	0.50	0.50	1.00	1.00	0.50
Plugins or Add-Ons	0.50	0.00	1.00	1.00	0.50	0.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00
Scalability	1.00	1.00	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	0.50	1.00	1.00
Testing	1.00	1.00	0.50	0.50	1.00	1.00	0.50	1.00	1.00	0.00	0.50	0.50	1.00
i18n and l10n	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	1.00
Validation	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.50
Multi-language Support (Groovy / Scala)	0.50	0.50	1.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	0.50
Quality of Documentation/Tutorials	0.50	1.00	0.50	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Books Published	1.00	1.00	0.50	1.00	0.50	0.50	1.00	1.00	1.00	1.00	0.50	0.50	0.00
REST Support (client and server)	0.50	1.00	0.50	0.00	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	0.50
Mobile / iPhone Support	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	1.00
Degree of Risk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.50	0.50
Totals	14.5	17	15	13.5	15	14	17	17.5	17	13.5	15.5	11.5	14

Ratings Logic

http://bit.ly/jvm-webfwk-ratings-logic

Top at TSSJS	Top at RWX 2010	Top at Devoxx 2010
Grails	Grails	GWT
GWT	GWT	Rails
Rails	Rails	Spring MVC
Spring MVC	Spring MVC	Grails
Vaadin ◁	Tapestry / Vaadin ◁	Wicket / Struts 2

Kuva 1. Web-ohjelmistokehysten vertailua valittujen kriteereiden perusteella. /4/

Matrix | **Weighted**

Weight	Criteria	Struts 2	Spring MVC	Wicket	JSF	Tapestry	Stripes	GWT	Grails	Rails	Flex	Vaadin	Lift	Play
10	Developer Productivity	5.00	5.00	5.00	5.00	10.00	5.00	10.00	10.00	10.00	0.00	10.00	5.00	10.00
0	Developer Perception	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Learning Curve	5.00	5.00	2.50	2.50	2.50	5.00	5.00	5.00	5.00	5.00	5.00	2.50	5.00
5	Project Health	2.50	5.00	5.00	5.00	2.50	2.50	5.00	5.00	5.00	2.50	5.00	5.00	5.00
5	Developer Availability	2.50	5.00	2.50	5.00	5.00	2.50	5.00	2.50	5.00	5.00	2.50	0.00	2.50
0	Job Trends	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	Templating	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	Components	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Ajax	2.50	5.00	2.50	2.50	2.50	2.50	5.00	2.50	2.50	2.50	5.00	5.00	2.50
5	Plugins or Add-Ons	2.50	0.00	5.00	5.00	2.50	0.00	5.00	5.00	5.00	5.00	5.00	2.50	5.00
10	Scalability	10.00	10.00	5.00	5.00	5.00	10.00	10.00	5.00	5.00	5.00	5.00	10.00	10.00
10	Testing	10.00	10.00	5.00	5.00	10.00	10.00	5.00	10.00	10.00	0.00	5.00	5.00	10.00
0	i18n and l10n	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	Validation	5.00	5.00	5.00	2.50	5.00	5.00	5.00	5.00	5.00	5.00	5.00	2.50	2.50
10	Multi-language Support (Groovy / Scala)	5.00	5.00	10.00	10.00	10.00	10.00	0.00	10.00	0.00	0.00	10.00	0.00	5.00
10	Quality of Documentation/Tutorials	5.00	10.00	5.00	5.00	5.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
0	Books Published	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	REST Support (client and server)	5.00	10.00	5.00	0.00	5.00	5.00	5.00	10.00	10.00	5.00	5.00	5.00	5.00
10	Mobile / iPhone Support	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	5.00	10.00	10.00	10.00
0	Degree of Risk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	**Weighted Totals**	70	85	67.5	62.5	75	77.5	80	90	82.5	50	82.5	62.5	82.5

Top JVM Web Frameworks (Weighted)

Grails
Spring MVC
Rails / Vaadin / Play ◁
GWT

Kuva 2. Sama vertailu painotettujen kertoimien mukaan. /4/

2.2.1. Vertailun tulokset ja johtopäätökset

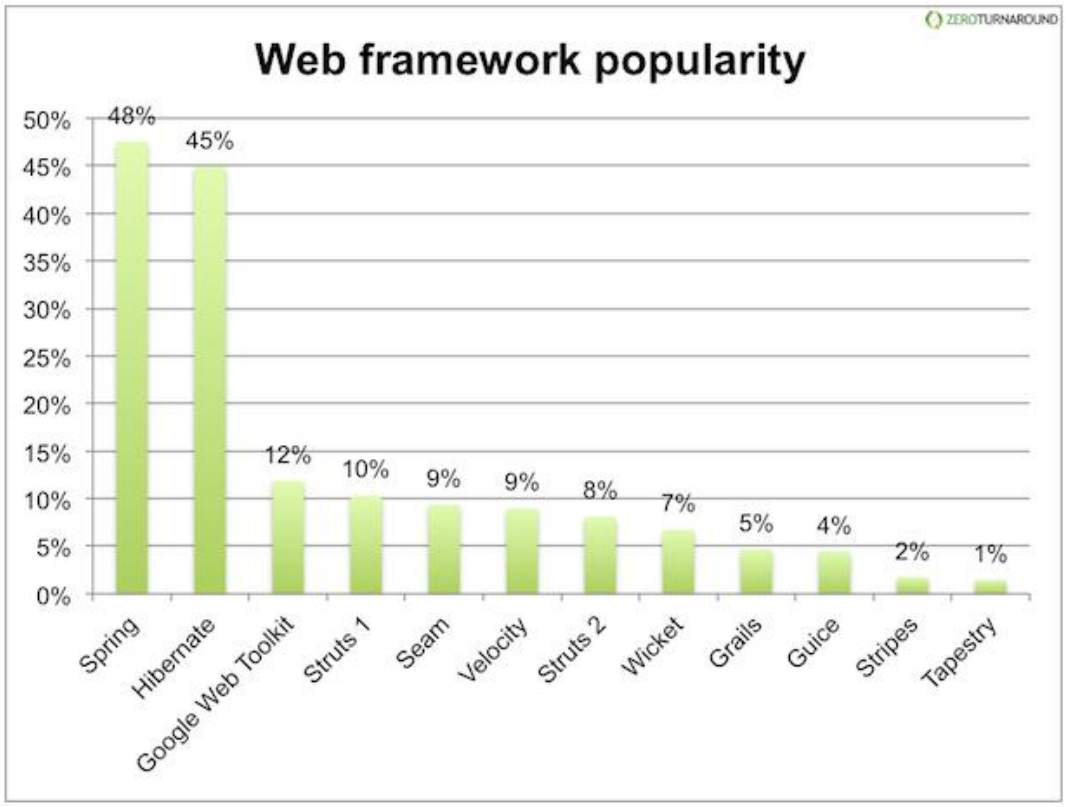
Vertailu osoittaa hyvin, mitä ominaisuuksia sovelluksilta vaaditaan ja mitä niistä yleisesti eniten arvostetaan (painotettu kerroin). Vaadin sijoittuu molempien vertailujen

pisteytyksillä mitattuna kärkisijoille. Raiblen esityksen jatkosivuilla vertailun painopiste keskittyy enemmän (amerikkalaisiin) projekteihin:

- Spring MVC ja Grails (VMWare)
- Wicket ja Tapestry (Apache)
- Ruby on Rails (Rails Core Team)
- GWT (Google).

Valitettavasti Vaadin ei vielä pärjää kovin hyvin monellakaan jatkovertailun parametreilla mitattuna ja kaavioita voi käydä katsomassa lähteen osoitteesta. Raible on kuitenkin listannut vertailun parhaimmistosta plussia ja miinuksia. Vaadin saa plussaa mm. GWT:n rajapinnan käyttämisestä sovelluskehitykseen, erinomaisesta tuesta teemoille ja pohjille, sekä vireästä yhteisöstä ja sen tuesta yritykselle. Vaadin onkin julkistanut eniten päivityksiä ja parannuksia ohjelmistoonsa vuoden 2010 aikana jatkovertailussa olleista ohjelmistokehyksistä. Miinusta Vaadin saa toisaalta laajasta muistin varauksesta, koska tilat varastoidaan istuntoihin (Sessions). /9/

Vertailuun on liitetty mukaan kaavio 2010 suosituimmista (laajimmalle levinneistä) Java-pohjaisista web-ohjelmistokehyksistä. Lähteenä on käytetty virolaisen ZeroTurnaround yrityksen julkaisemaa vertailua. Itse asiassa kyseisen yrityksen tuote JRebel muistuttaa monilta ominaisuuksiltaan Vaadinta. Ja jotta tämä paisuva ohjelmistokehysten vertailukirjo saisi ansaitsemansa päätöksen, suosituimpien Java-pohjaisten ohjelmistokehysten kärkikasti ei olekaan òihanò sama kuin Raible on omassa vertailussaan halunnut esittää. Tosin ZeroTurnaroudin tutkimukset perustuvat Java-kehittäjille lähetettyihin kyselyihin ja yrityksen kotisivuilta löytyy paljon muitakin mielenkiintoisia kyselyitä sekä kaavioita aiheeseen liittyen. /9/, /14/



Kuva 3. Suosituimmat Java-pohjaiset web-ohjelmistokehykset erään ohjelmisto-kehittäjän tekemän kyselyn mukaan. /14/

3. VAADIN

Vaadin on suomalainen Java-pohjainen ohjelmistokehys, joka käyttää Ajax-tekniikkaa dynaamisten web-palveluiden ja RIA-sovellusten toteuttamiseen. Tuotekehitys alkoi IT Mill Oy:n toimesta ja yritys tunnettiin erilaisten yritysten web-käyttöliittymien teknologian kehittäjänä. Yhtiön uusi nimi on muuttunut Vaadin Oy:ksi lippulaivansa Vaadin-sovelluksen myötä. Vaadin on parin viimeisen vuoden aikana noussut nopeasti kansainväliseen julkisuuteen. Tämän hetkisen arvion mukaan tuotetta käyttävistä ohjelmistokehittäjistä 95 prosenttia on Suomen ulkopuolelta yli sadassa maassa, toteaa Vaadin Oy:n toimitusjohtaja Joonas Lehtinen lehtiartikkelissa. /7/

3.1. Kehityshistoria

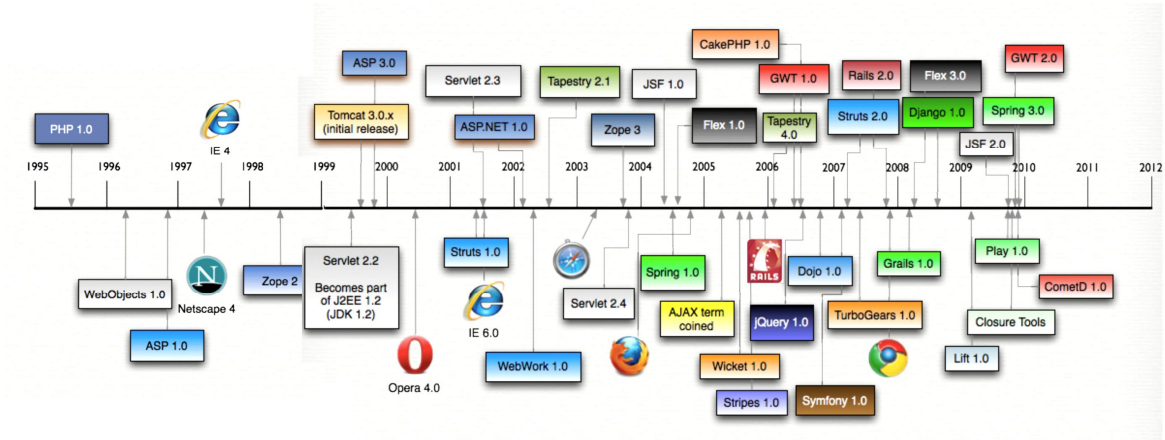
IT Mill perustettiin vuonna 2000, kun ryhmä suomalaisia ohjelmistokehittäjiä halusi tehdä markkinoiden parhaan työvälineen web-selaimella käytettävien yrityssovellusten rakentamiseen. Ensimmäinen projekti tehtiinkin suurelle kansainväliselle lääkealan yritykselle vuonna 2001 ja se on edelleen käytössä. Tuolloin sovelluksen nimi oli Millstone Library. Millstonen 3-versio julkaistiin vuonna 2002 osin avoimen lähdekoodin web-ohjelmointikehyksenä. Yksinkertaista Ajax-tekniikkaa sisällytettiin asiakaspuolen (client-side) toteutuksiin kaupallisessa versiossa ja kehitystyö jatkui tekniikan kehittyessä koko ajan. /2, s. 5/

Seuraavan sukupolven ohjelmointikehyksen nimi muuttui; IT Mill Toolkit Release 4 esitteli uuden kehittyneemmän Ajax-pohjaisen tekniikan sovellusten esittämiseen asiakas/palvelin ï mallissa vuonna 2006. Jo seuraavan vuoden 2007 lopulla julkaistiin IT Mill Toolkit 5, joka sisälsi merkittäviä muunnoksia ja otti lopullisen askeleen kohti Ajax-tekniikkaa. Asiakaspuolen käyttöliittymä kirjoitettiin kokonaan uusiksi käyttäen GWT (Google Web Toolkit) työkaluja. Tämä mahdollisti Java-kielen käyttämisen sekä asiakas- että palvelinpuolella, joka toisaalta helpotti komponenttien käyttöä, integrointia ja muokkausta. GWT:n mukaantulo ei aiheuttanut mitään muutoksia palvelinpuolen (server-side) ohjelmointirajapinnassa, koska GWT on osa ns. piilotettua selaintekniikkaa ja se tukee laajasti eri selaimia. /2, s. 5/

Muitakin merkittäviä muutoksia tehtiin 5-version aikana. Yksi niistä oli ohjelman julkaisu avoimen Apache-lisenssin alla, jotta nopeampi yhteisön tuoma kehitys olisi mahdollista. Vakaa versio 5.3.0 julkistettiin maaliskuussa 2009, kunnes jo pian sen jälkeen IT Mill Toolkit nimettiin Vaadin Framework -nimiseksi, tai lyhyemmin pelkästään Vaadin. Yhtiön nimi muuttui samalla Vaadin Oy:ksi. /2, s. 6/

Ohjelmointikehyksen versioksi muotoutui Vaadin 6 ja siinä oli mukana integroidun kehitysympäristön Eclipsen liitännäinen (Plugin) ja samalla kokeellinen visuaalinen (WYSIWYG) editori. Uusin ohjelmaversio on tällä kirjoitus hetkellä 6.7.2 ja 7-versio on jo tulossa. Kehittäjien yhteisö on ollut aktiivinen ja ominaisuuksia luvataan lisää uuden kehittyvän teknologian myötä.

Kuva 4. Vaadin-ohjelmistokehyksen kehitys aikajanalla merkittävimpien muutosten mukaan. /5/



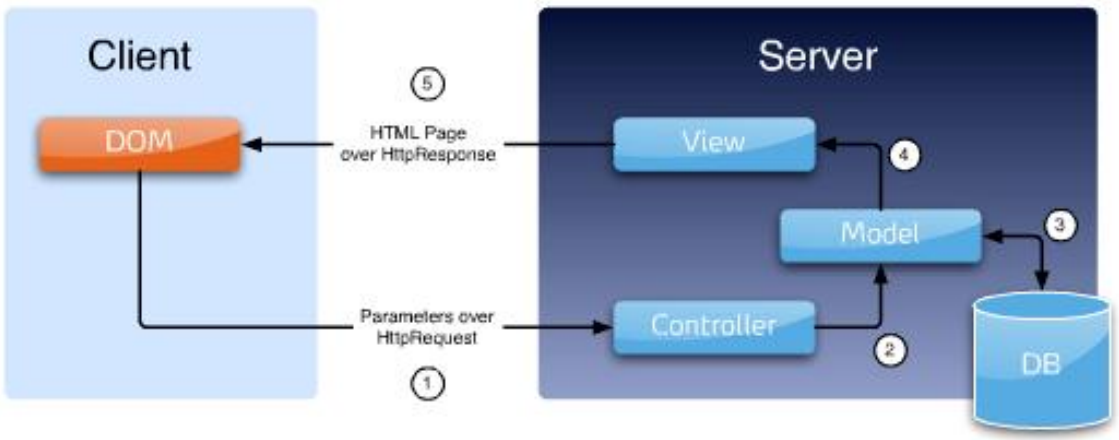
Kuva 5. Muiden merkittävimpien web-ohjelmistokehysten ja niihin vaikuttaneiden ohjelmistojen kehitys aikajanalla. /8/

3.2. Toiminta-ajatus

Vaadin käyttää Javan kirjastoja ja työkaluja, joiden tarkoitus on tehdä helpommaksi korkealaatuisten käyttöliittymien tekeminen ja niiden ylläpito. Javan ohjelmointirajapinta mahdollistaa metodikutsut eri virtuaalikoneissa sijaitsevien olioiden välillä ja tarjoaa abstraktiotason, jota käyttävän ohjelmoijan ei tarvitse syvällisesti pohtia hajautuksen monimutkaisuutta ja potentiaalisia virhetilanteita.

Kun perinteinen web-ohjelmointi vaatii usein asiakaspuolen hallinnalta monia sovelluskohtaisia määrittelyjä rajapinnoissa kommunikoidakseen palvelinpuolen kanssa, keskittyy Vaadin lähinnä käyttöliittymän hallintaan ja Ajax-kommunikaatioon asynkronisesti selaimen (asiakas) ja palvelimen välillä. /2, s. 1-3/

Tällaisen palvelinpuolen arkkitehtuurin ja ohjelmointimallin etu on, ettei ohjelmoijan tarvitse miettiä tai opiskella selaintekniikkaan liittyviä kieliä, kuten HTML tai JavaScript. Suurin osa ohjelman logiikasta ajetaan palvelimella ja Ajax-tekniikkaa käytetään monipuolisesti hyväksi. Tämä mahdollistaa vuorovaikutteisen sovelluksen toiminnan òrikkaampanaò kuten perinteisen työpöytäsovelluksen tapana on toimia (RIA-sovellus). Vaadin hyödyntää tässä toiminnassa, muista vastaavista kehyksistä hieman poiketen, suoraan integroituja GWT:n työkaluja ja kirjastoja. /2, s. 1-3/



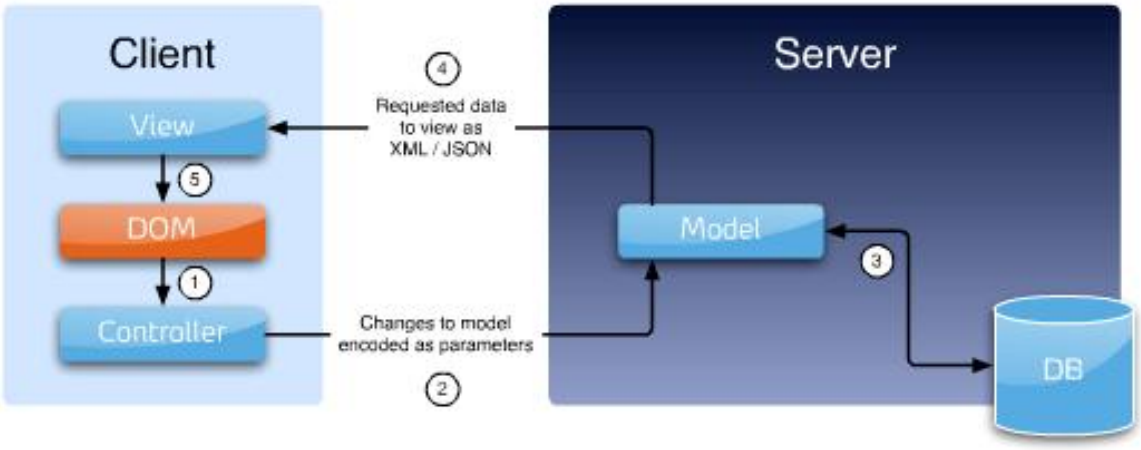
Kuva 6. Perinteinen, yksinkertainen MVC-malli asiakas/palvelin-vuorovaikutuksesta. /5/

MVC-arkkitehtuurissa ohjelma jaetaan kolmeen osaan: malliin (Model), näkymään (View) ja käsittelijään (Controller). Web-sovelluksiin MVC-arkkitehtuurityyli sopii hyvin.

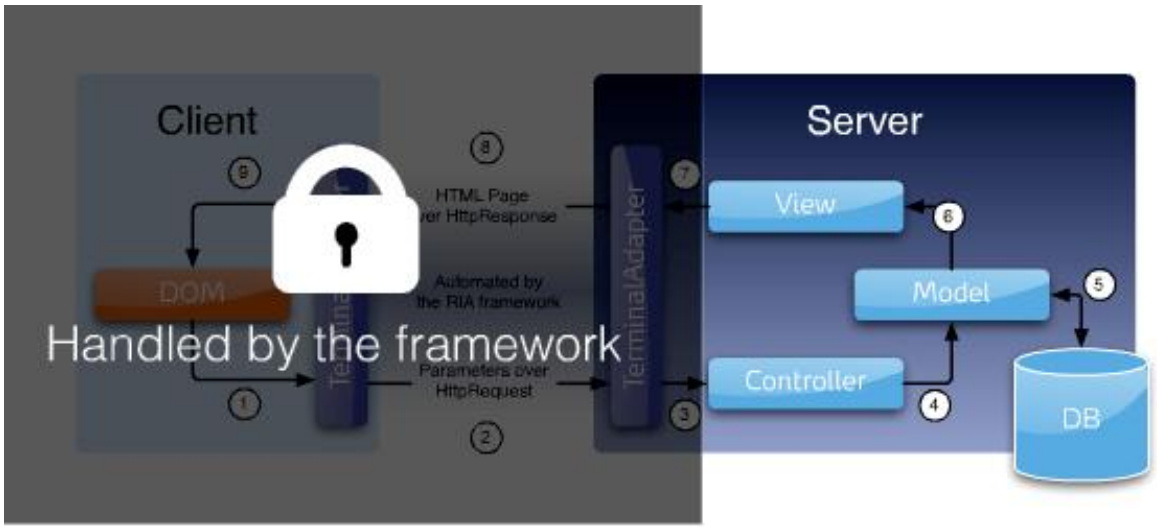
- Malli toimii tietokantana.
- Näkymä kertoo sovelluksen tilan ja mahdollistaa lisäviestit (HTML).
- Kontrolleri käsittelee mallia ja valitsee näkymän.

MVC-ratkaisuissa käytetään termejä Push ja Pull, riippuen siitä, millä tapaa web-kehyksessä on haluttu organisoida pyynnöt ja vastaukset. Push on pyyntöperustainen ratkaisu, jossa kontrollerit vastaavat suoraan asiakkaan tekemiin pyyntöihin. Varsinainen pyynnön ohjauksen mekanismi vaihtelee kehyksittäin. Tällaista mallia käyttävät mm. Spring MVC ja Grails. /10/

Pull-ratkaisu on komponenttimalli, jossa ohjelmistokehykset kääntävät pyyntö-vastaus-mallin toisin päin. Näissä kehyksissä logiikka on käytettävissä uudelleen komponenteissa, joista uusin päivitetty tieto on saatavilla. Yksittäiset pyynnöt abstrahoidaan pois ja näkymä on vastuussa mallin kutsumisessa. Tällaista mallia käyttävät mm. Vaadin ja Wicket. /10/



Kuva 7. Esimerkki RIA-sovellusten toiminnasta asiakaspuolen MVC-mallissa. /5/



Kuva 8. Vastaava palvelinpuolen MVC-malli, jossa on kuvattu myös ohjelmisto-kehyksen (Vaadin) toiminta RIA-sovelluksissa suoraan palvelimen kanssa. /5/

3.3. Tavoitteet

Kaikilla ohjelmistojen ja sovellusten kehittäjillä lienee ykköstavoitteenaan luoda ohjelmansa maailman suosituimmaksi ja strategiaansa parhaiten soveltuvaksi. Vaadin on luetellut ohjekirjassaan muutamia omaan filosofiaansa sopivia tavoitteita.

Oikea työkalu oikeaan tarkoitukseen

Vaadin on suunniteltu RIA-tyyppisten web-sovellusten tekemiseen, ei web-sivustojen tekemiseen. /2, s. 5/

Yksinkertaisuus ja ylläpidettävyys

Tekijätiimi haluaa korostaa sovelluksen tehokkuutta, yksinkertaisuutta ja ylläpidettävyyttä. Tästä syystä on keskitytty paljon kehyksen käyttöliittymään ja yritetty saada toteutettua paras mahdollinen, tarkoitukseensa sopiva ratkaisu. /2, s. 5/

XML ei ole suunniteltu ohjelmointia varten

Web-teknologiat ovat pitkälti dokumenttikeskeisiä ja täten melko rajoitettuja käyttöliittymänsä sivunkuvauskieleen (esimerkiksi HTML). XML-kieli ei kuitenkaan ole tarkoitettu sivunkuvauskieleksi, vaan sillä kuvataan tiedon (data) rakenne ilman ennalta määrättyjä koodeja. Vaadin keskittyy mieluummin monipuolisten ja joustavien käyttöliittymien tekemiseen ja ohjelmoimiseen, kuin pelkästään sisällyttää niitä joihinkin valmiisiin ja toiminnaltaan rajoitettuihin pohjiin (template, layout). /2, s. 5/

Työkalujen ei tulisi rajoittaa työskentelyä

Ohjelmointikehyksen käyttöliittymällä pitäisi voida toteuttaa palveluja komponenttien avulla siten, että haluttu toiminnallisuus on mahdollista ilman rajoituksia. Oli sitten kyseessä uuden komponentin tekeminen, lisääminen, muokkaaminen, uudelleen käyttö ja ylläpito, tai vaikka kolmannen osapuolen komponenttien avulla toteutettu toimintojen laajennettavuus (esim. GWT/widget). /2, s. 5/

4. VAADIN ARKKITEHTUURI

Vaadin koostuu palvelinpuolen kehyksestä ja asiakaspuolen moottorista (Client-side Engine). Ohjelmat ajetaan selaimessa Javascript-kielisinä, joka on selainpohjainen skriptikieli. Käännös Java-kielestä skriptikieleksi tehdään GWT:n kääntäjän avulla.

4.1. Ajax

Asynkroninen viestintä mahdollistaa keskustelun palvelimen kanssa taustalla niin, että tietty osa sivusta voidaan päivittää ilman, että koko sivua tarvitsee ladata uudelleen. Tämä lisää käytettävyyttä sekä nopeutta käyttäjän ja palvelimen välillä, jolloin verkkosivuista tulee yhä enemmän työpöytämäinen interaktiivinen verkkosovellus.

Vaadin käyttää Ajax-pohjaista tekniikkaa ja sitä kautta asynkronista viestintää asiakkaan (selaimen) ja palvelimen välillä. Ajax ei varsinaisesti ole tekniikka itsessään, vaan se on yhdistelmä eri teknologioita: XHTML, CSS, DOM, JavaScript, XMLHttpRequest ja XML. /2, s. 38/

Eri tekniikoiden toiminnot Ajax-pohjaisessa kommunikaatiossa on jaettu seuraavasti:

- XHTML-kieltä merkintäkielenä.
- CSS-tyyliohjeita muotoilukielenä.
- XML:ää tiedonsiirtoformaattina palveluiden välisten, dokumenttimuotoisten sanomien rakenteen kuvaamiseen.
- DOM-rajapintaa sisällön muokkaamiseen, tallentamiseen ja näyttämiseen (X(HTML) + JavaScript).
- XMLHttpRequest-luokkaa asynkronisen viestin lähetykseen ohjelmointirajapinnassa.
- JavaScript-kieltä sitomaan yllämainitut teknologiat yhteen toimintojen dynaamisuuden lisäämiseksi.

4.2. JSON ja GWT

Vaadin on käyttänyt tiedonsiirtoon ja -vaihtoon myös JSON -tekniikkaa 5-versiostaan alkaen. Se on helppokäyttöinen ja ennen kaikkea tehokas tapa vuorovaikutukseen JavaScript-pohjaisissa teknologioissa, kuten Ajaxissa. JSON on XML:ää yksinkertaisempi, selkeämpi ja kevyempi formaatti, ja näin ollen se soveltuu hyvin tiedonsiirtoon. Formaatin käyttöön lienee myös syynä GWT:n mukaan tulo osaksi Vaadin-ohjelmointikehystä.

Asiakaspuolen moottori (Client-Side Engine) hyödyntää JSON-tekniikkaa GWT:n kautta. GWT tukee JSON-tekniikalla tapahtuvaa viestintää täysipainoisesti JSON-kirjastonsa avulla, vaikkakin itse JSON:in toteutus ja toiminta ovat käytännössä täysin näkymättömiä käyttäjälleen. /2, s. 36-39/

Käyttöliittymän näkymät ja niiden muutokset web-sivulla on toteutettu terminaalisovittimen (Terminal Adapter) avulla ja sen kautta toteutetaan asynkroniset HTTP(S) palvelupyynnöt. Tähän kommunikointiin käytetään erillistä JSON-pohjaista UIDL-kieltä, jonka tarkoituksena on kuvantaa palvelimen vastaukset selaimelle. Käytännössä UIDL viestit jäsennetään selaimessa ja käännetään GWT komponenteiksi (tai GWT:n terminologian mukaan widgeteiksi, suom. vimpain). /2, s. 36-39, s. 309/

4.3. Komponentit

Komponenteilla tarkoitetaan ohjelmiston valmiita toiminnallisia lisäosia, esimerkiksi painikkeita, listoja, ikkunoita tai animoituja kuvakomponentteja. Oliopohjaisissa kielissä komponentit järjestetään yleensä hierarkkisiksi luokkakirjastoiksi, joiden avulla voidaan suorittaa haluttuja toimintoja rajapintojen välillä.

Sisällön tallennus ja siirtäminen on eräs komponenttien tärkeimmistä tehtävistä. Toinen on komponenttien uudelleen käyttö siten, että ohjelmoijat voivat suunnitella ja toteuttaa niitä käytettäväksi muissakin ohjelmissa, käyttöympäristöstä riippumatta. Komponentteja voi kutsua myös òrakennuspalikoiksiò, joita käytetään hyväksi luotaessa ohjelmia tai web-sovelluksia.

Vaadin tarjoaa kattavan valikoiman valmiita käyttöliittymän komponentteja ja sallii myös komponenttien tekemisen ja muokkaamisen omiin tarpeisiin. Sisäänrakennettujen vaihtoehtojen lisäksi on tarjolla myös muiden riippumattomien lähteiden valmistamia komponentteja: maksullisia ja ilmaisia.

Vaadin arkkitehtuurin mukaan jokaisella palvelinpuolen komponentilla on vastineensa asiakaspuolella. Vuorovaikutteiset tapahtumat komponenttien välillä toteutetaan HTTP-palvelimen ja terminaalisovittimen kautta. Käyttäjän interaktio komponenttien kanssa luo tapahtumia (Events), jotka ensin prosessoidaan asiakaspuolella GWT:llä Javascriptiksi ja sen jälkeen tiedot välitetään käyttöliittymän sovelluslogiikalle (Application UI Logic). Vastaavasti jos sovelluslogiikka tekee muutoksia käyttöliittymän komponentteihin, terminaaliadapteri generoi vastauksen tekemällä muutoksen selaimessa. /2, s. 36, s. 66/

4.4. Teemat

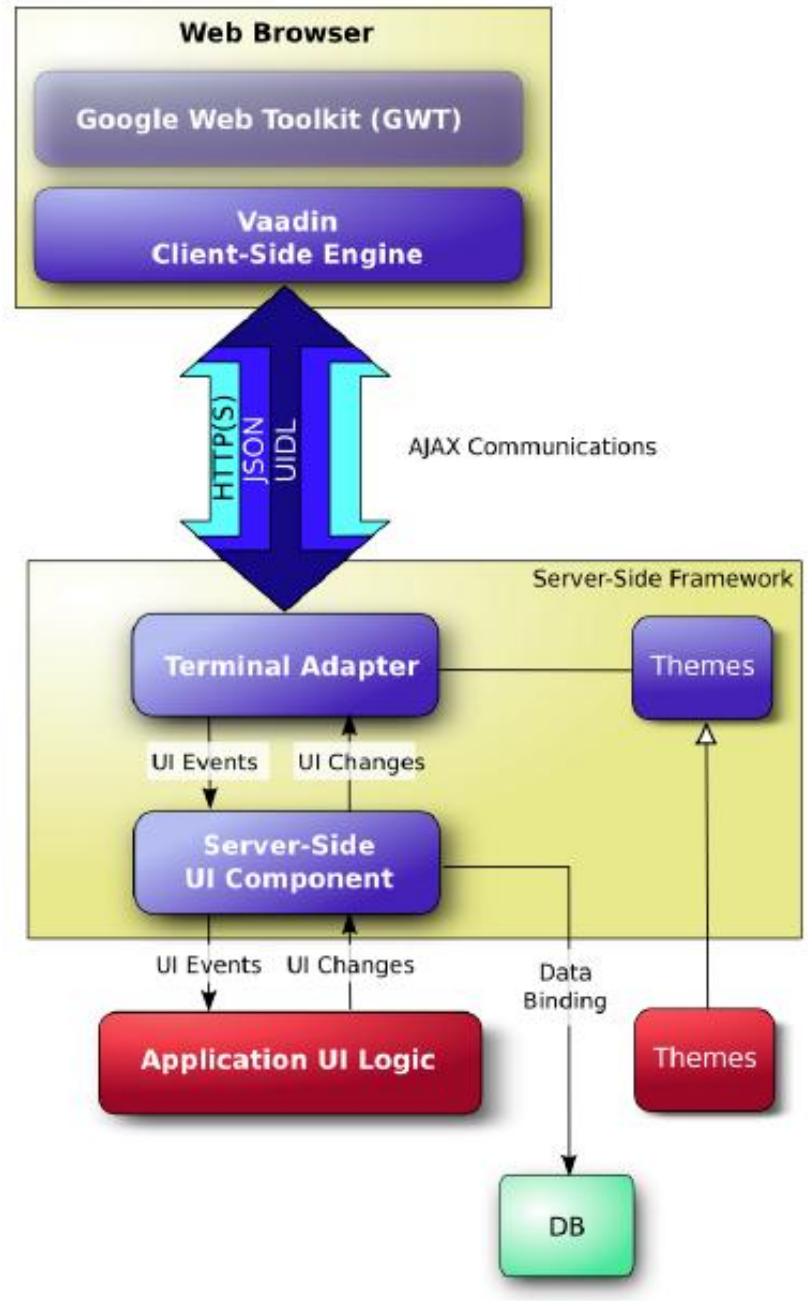
Vaadin erottelee käyttöliittymän ulkoasun käyttöliittymän logiikasta teemojen (Themes) avulla. Teemat kontrolloivat web-ohjelmien ja -dokumenttien visuaalista ilmettä ja sen hallintaan sekä muotoiluun käytetään yleisesti CSS-tekniikkaa. Tällä tekniikalla tarkoitetaan kokonaisuudessaan ohjelmaan liittyviä tyyliohjeita ja toisaalta tyylikieltä, jonka perussyntaksi muodostuu valitsimesta, ominaisuudesta ja arvosta. /2, s. 37/

Teemat voivat sisältää tyylitiedostojen (style.css) lisäksi käyttäjän omia, räätälöityjä HTML-pohjia ja mitä tahansa tarpeellista grafiikkaa. Normaalisti pakettiin on sisällytettynä myös kaksi sisäänrakennettua, omaa alkuperäistä Vaadin-teemaa: Reindeer ja Runo. /2, s. 37/

4.5. Datan liittäminen

Vaadin käyttää oliopohjaista mallia datan liittämiseen käyttöliittymän komponentteihin siten, että yhteen sopimaton data on mahdollista käyttää ja päivittää suoraan ohjelmassa, ilman kontrolliin tarvittavaa koodausta. Tällä tavoin Vaadin voi käyttää òvirtuaalista oliotietokantaaò sisältäpäin. Tämäntyyppisessä ORM-mallissa oliot, luokat ja periminen on

suoraan tuettuja sisäisesti, vaikka komponentteja voidaankin sitoa myös muihin ulkoisiin tietokantoihin. Suurin osa komponenteista on sidoksissa tietokantoihin joka tapauksessa. /2, s. 37-38, s.207/



Kuva 9. Vaadin-arkkitehtuuri kaaviona. /2, s. 36/

5. SOVELLUKSEN ARKKITEHTUURI

Vaadin-ohjelmointikehyksellä tehty sovellus ajetaan Java-sovelmana (Java Servlet) Java-sovelluspalvelimessa, esimerkiksi Tomcatissa. Sovelmaa eli pientä Java-kielistä ohjelmaa voidaan kutsua myös luokkatiedostoksi. Sovellusluokka (Application Class) toimii aloituspisteenä Servlet-rajapinnassa (Java Servlet API), jonka päällä Vaadin suorittaa periaatteessa kaikki toimintonsa. Tämä luokkatiedosto palvelee Ajax-palvelupyyntöjä (Request) ja sillä luodaan sekä hallitaan käyttöliittymän komponentteja. Alimman kerroksen tasolla Vaadin käyttää Servlet-rajapinnan kautta terminaalisovitinta vastaamaan ns. säiliön palvelupyyntöihin. /2, s. 35/

5.1. Servlet-säiliö

Servletit laajentavat palvelimen toiminnallisuutta ja lisäksi ne mahdollistavat monien yhtäaikaisten pyyntöjen suorittamisen. Servlettien käyttö vaatii ohjelmistolta Servlet-säiliön (Servlet Container), joka käytännössä tarkoittaa Java-sovelluspalvelinta.

Säiliön tehtävänä on tarjota suoritusympäristö komponenteille ja olla vastuussa servlettien elinkaaresta. Se suorittaa URL-lähteen kuvauksen ja varmistuksen palvelunpyytäjän oikeuksista. Tämän jälkeen säiliö luo instanssin Vaadin-kehyksen ApplicationServlet -luokasta, joka perii Servlet-rajapinnan HttpServlet ï luokan. Luokka seuraa istuntoja (Sessions) HttpSession ï liitäntää hyväksi käyttäen ja yhdistää instanssin jokaiseen istuntoon. Jokaisen istunnon elinkaaren aikana, ohjelmointikehys välittää tietoa käyttäjän toiminnoista asianomaiselle sovelluksen instanssille ja edelleen käyttöliittymän komponentille. /2, s. 39/

5.2. Ikkunat

Sovelluksessa täytyy ihan aluksi määritellä sovellusluokka (Application Class), joka perii abstraktin com.vaadin.Application ï luokan, joka toteutetaan init () ï metodilla.

Sovellusluokan tehtävänä on tarjota myös palveluja ikkunoiden käsittelyyn, kontrollin toteutukseen ja teeman valintaan. /2, s. 44/

Alustuksessa tärkein tehtävä on luoda pääikkuna (Main window), jollainen on jokaisella sovelluksella. Toinen minimivaatimus sovellukselle on sovelluksen käyttöönotto servlettinä Servlet-säiliössä. /2, s. 44-46/

```
package com.vaadin.ui.*;

import com.vaadin.ui.Label;
import com.vaadin.ui.Window;

public class HelloWorld extends com.vaadin.Application {

    public void init() {

        // Main window is the primary browser window
        final Window main = new Window("Hello window");
        setMainWindow(main);

        // "Hello world" text is added to window as a Label component
        main.addComponent(new Label("Hello World!"));
    }
}
```

5.3. Tapahtuman kuuntelija

Kuten aiemmin todettiin, käyttäjän interaktio komponenttien kanssa luo tapahtumia (Events). Monissa Java-pohjaisissa ohjelmistokehyksissä noudatetaan samaa kaavaa käyttäjän syöttöjen ja sovelluslogiikan viestinnän välillä. Tällainen nk. tarkkailumalli sisältää kahdenlaisia elementtejä; olion ja joukon tarkkailijoita, jotka òkuuntelevatò tapahtumia olioihin liittyen. Useimmissa tapauksissa tarkkailijoita on vain yksi kutakin tapahtumaa kohden, sovelluslogiikan määrittelemänä. Näitä tarkkailijoita kutsutaan kuuntelija-olioiksi (Listeners). /2, s. 41/

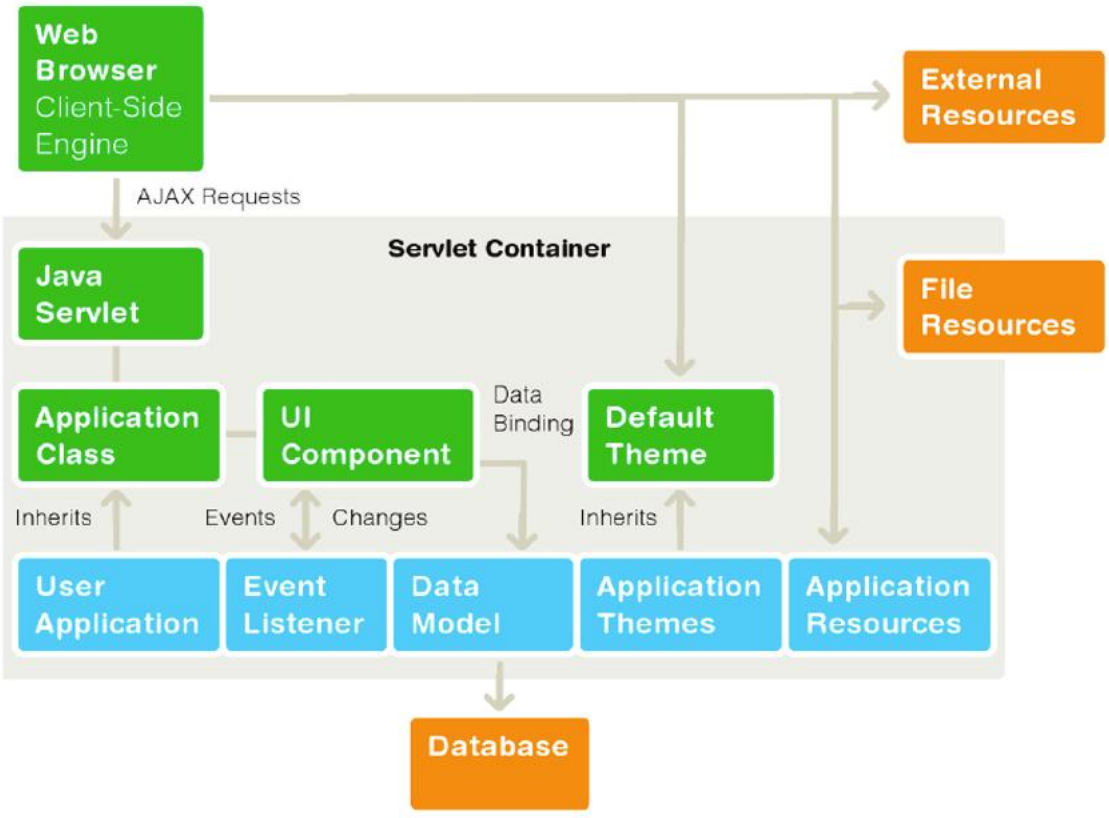
Tällaisten kuuntelija-olioiden käyttö edellyttää kuuntelija-luokan määrittelyä, jossa toteutetaan kuuntelijarajapinta. Vaadin käyttää kuuntelijoiden hallintaan AbstractComponent ï luokkaa, joka on perusluokka kaikissa käyttöliittymän komponenteissa. Tämä tarkoittaa sitä, että tapahtumia voidaan kuunnella minkä tahansa

komponentin suhteen. Kuuntelijat ovat rekisteröity komponenteissa addListener()-metodilla. Kun kuuntelijan metodia kutsutaan, välitetään sille tapahtuma-olio, joka sisältää tietoa tapahtumasta. /2, s. 41/

5.4. Resurssit

Käyttöliittymä voi näyttää kuvia tai sillä voi olla linkkejä web-sivuihin ja ladattaviin dokumentteihin. Näitä kutsutaan lähteiksi (Resources), jotka voivat olla ulkoisia tai palvelimen tarjoamia, sekä sisäisiä, itse sovelluksen tarjoamia lähteitä. /2, s. 45/

Vaadin tarjoaa lähteen haun sovellusten rajapinnassa siten, että palautuksena saadaan useita erityyppisiä lähteitä; tiedostoja tai dynaamisesti luotuja lähteitä. Nämä sisältävät StreamResource ï luokan ja URI:n sekä parametrin käsittelijät. /2, s. 53/



Kuva 10. Vaadin-ohjelmistokehyksellä tehdyn sovelluksen arkkitehtuuri. /2, s. 44/

6. OHJELMISTON ASENNUS JA TESTAUS

Book of Vaadin -kirjan ohjeiden mukaan ohjelmiston asennus on suhteellisen helppoa yleisimpiin käyttöjärjestelmiin. Helpointa se ainakin näyttäisi olevan Windows-käyttöjärjestelmään, ilman enempää komentokielen käyttämistä ohjelmien asennuksessa (vrt. Linux/UNIX).

Ohjelmistojen ja kehitysympäristöjen vaihtoehtoja on esitelty tarkemmin Vaadin-kotisivun ominaisuuksien esittelyssä; https://vaadin.com/features. Tästä huolimatta voi olla järkevintä asentaa ja käyttää ohjelmistoista niitä versioita, joita ohjekirjassa on suositeltu asennettavaksi.

Joitain ongelmia voi tulla yhteensopivuuksien kanssa, elleivät esimerkiksi ohjelmien tietyt (uusimmat) versiot tue keskenään toisiaan. Siinä tapauksessa on syytä tarkistaa ja varmistaa kaikkien asennettavien ohjelmien dokumentaatiot.

Asennettavat ohjelmat ovat:

- Vaadin (tai pelkkä Vaadin Plugin, liitännäinen, käytettävään IDE-ohjelmaan)
- IDE-ohjelma
- Java-ohjelmisto
- Java-palvelinohjelma

6.1. Vaadin ï asennuspaketti vai pelkkä Plugin?

Vaadin ei ole sidoksissa mihinkään tiettyyn IDE-ohjelmaan, vaikka asennuksessa suositellaan Eclipse IDE-ohjelman käyttöä. Tukea löytyy myös muille vastaaville ohjelmille, mutta laajimmin sitä löytyy Eclipselle.

Asennus onnistuu molemmilla tavoilla, joko asentamalla Vaadin-paketin suoraan sivustolta tai asentamalla pelkän liitännäisen suositeltuun IDE-ohjelmaan. Nopein tapa on pelkän liitännäisen asennus esim. Eclipse IDE-ohjelmaan, koska käytettäessä kehitysympäristöä, joutuu IDE-ohjelman joka tapauksessa asentamaan. Molempien vaihtoehtojen asennus ei

ole pahitteeksi siinäkään tapauksessa, mikäli haluaa tutkailla Vaadin-paketin sisältöä ja testata siellä olevia demo-sovelluksia.

6.2. Suositellun kehitysympäristön käyttöönotto

Vaadin tukee laajasti eri vaihtoehtoja kehitysympäristön rakentamiseen. Yhteensopivia ohjelmia on esitelty kotisivulla ja sen mukaan jokainen voi valita mieluisimman ratkaisun. Aloittelijan kannattaa kuitenkin tukeutua ohjekirjassa suositeltuun kehitysympäristöön ja sen osien asennukseen vaihe vaiheelta. Siitäkin huolimatta, vaikka ohjekirjaa ei olisi päivitetty kaikilta osin vastaamaan kaikkien ohjelmistojen päivityskehitystä. Tällä varmistetaan kuitenkin kaikkien ohjelmien toimivuus keskenään, ilman mahdollisten uusimpien päivitysten tuomia ongelmia.

6.2.1. Järjestelmävaatimukset

Vaadin on Java-pohjainen ohjelmistokehys, joten se toimii kaikissa yleisimmissä käyttöjärjestelmissä, jotka tukevat Java-ohjelmistoalustaa (5 tai uudempaa versiota). Käyttöjärjestelmistä yleisimmät ovat Windows, Linux ja UNIX (mm. Mac OS X ja Sun Solaris). Ohjeiden mukaan näyttäisi asennuksen aloitus olevan mutkattominta Windows-käyttöjärjestelmälle, ilman komentokielisiä käskyjä.

6.2.2. Java-ohjelmisto

Javan asennus voi näyttää monimutkaiselta, koska kyseessä on laajempi kokonaisuus eri teknologioita. Koko Java-kehitysympäristöstä puhuttaessa, käytetään termiä Java SDK. Paketti sisältää kehitystyökalupaketin (JDK), johon kuuluu kääntäjä (javac) ja muut työkalut (jar, javadoc, jdb). Kehitysympäristö sisältää myös täydellisen ajoympäristön (JRE), joka tarvitaan käännettyjen ohjelmien ajamiseen.

Javan asennuksesta selviää periaatteessa yhdellä helpolla asennuksella. Ohjeiden linkin mukaisesti ladataan ja asennetaan Java Standard Edition (Java SE) 6 ï versio, joka sisältää JDK:n ja JRE:n.

Java Enterprise Edition (Java EE) on tarkoitettu palvelinsovellusten kehittämiseen ja ajamiseen ja sisältää mm. nimeämis- ja hakemistopalvelut (JNDI), komponenttirajapinnan (EJB), servlet-, portlet- ja JSP-määrittelyt sekä muita web-palvelintekniikoita. Java EE palveluiden käyttöönotto huomioidaan tässä tapauksessa Eclipse IDE-ohjelman asennuksessa. /12/

6.2.3. Eclipse IDE

Eclipse IDE-ohjelmasta suositellaan asennettavaksi ns. Genymede-versiota, joka on tarkoitettu Java EE kehittelyyn. Asennetun ohjelman kautta voi tarkistaa ja päivittää mm. Vaadin Plugin ja Vaadin kirjaston ennen työskentelyä sekä uuden projektin luomista.

Vaadin tarjoaa hieman laajempaa tukea Eclipse IDE-ohjelman käytölle. Tuki sisältää Vaadin demo-paketin asennuksen ja testauksen ilman erikseen asennettavaa palvelinta, sekä Vaadin Plugin asennuksen. Liitännäisen avulla Eclipsessä on mahdollista mm.

- luoda uusia Vaadin projekteja,
- luoda räätälöityjä teemoja,
- luoda asiakaspuolen widgetejä ja widget-kokoelmia,
- muokata komponentteja visuaalisesti WYSIWYG-editorilla,
- ja päivittää Vaadin kirjastoa helposti tarpeen mukaan. /2, s. 4/

6.2.4. Apache Tomcat ï palvelin

Apache Tomcat on kevyt Java-palvelin, joka soveltuu hyvin suositellun kehitysympäristön käyttöön. Muitakin vaihtoehtoja on palvelinohjelmalle, ja niitä löytyy Vaadin-kotisivuilta lueteltuna.

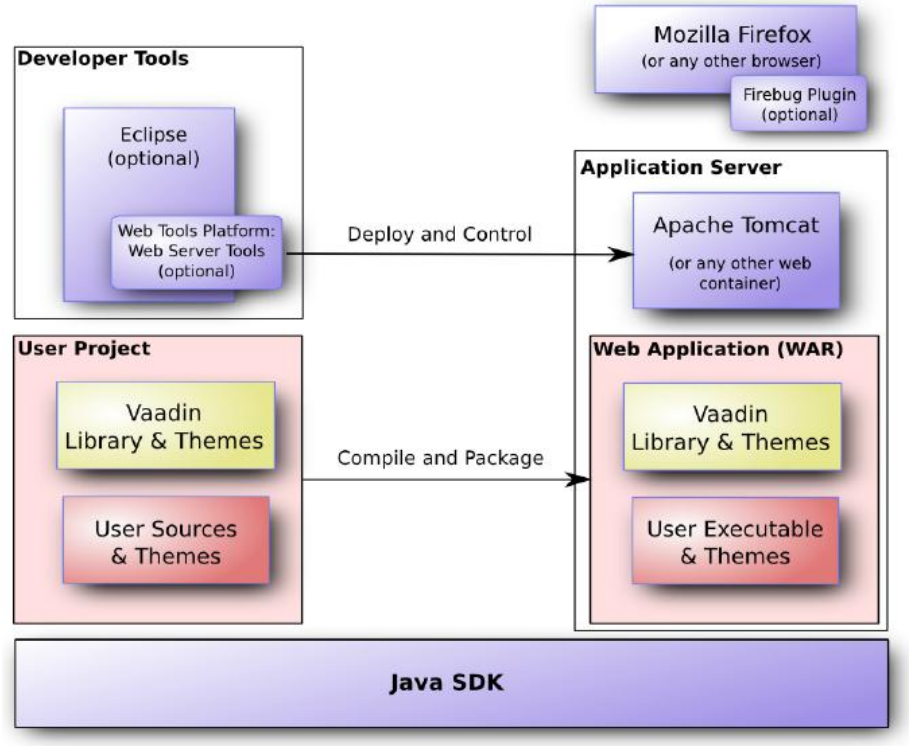
Ohjekirjan mukaan Apache Tomcat 6-versio (Core Binary Distribution) tulisi asentaa käyttäjän oikeuksilla silloin kun toimitaan paikallisesti omalla koneella (localhost), kuten tässäkin testauksessa tehtiin. Muussa tapauksessa asennus vaatii ylläpitäjän tai pääkäyttäjän oikeudet, esimerkiksi jos halutaan ottaa web-sovelluksia käyttöön laajemmin verkossa olevalla Tomcat-palvelimella. /2, s. 14/

Asennusvaiheen oletusportit ja määrittelyt antoivat herjan käyttöoikeuksista, mutta ohittamalla ne asennus kuitenkin onnistui. Näihin lienee syytä perehtyä tarkemmin siinä tapauksessa, jos aiotaan tehdä sovelluksia koko verkon käyttöön. Seikkaperäisemmät ohjeet löytyvät Apachen kotisivuilta ja Tomcat-dokumentaatiosta, jossa suositellaan myös Apache Ant (Java-kirjaston) asentamista samalla kertaa.

6.2.5. Selaimen valinta

Vaadin tukee useimpia selaimia ja niiden versioita. Jos kuitenkin halutaan luoda omia, räätälöityjä teemoja, muokattuja pohjia tai uusia käyttöliittymän komponentteja, Vaadin suosittelee Firefox-selaimen käyttöä, yhdessä Firebug-testauksen (debugger) kanssa. Vaadin lupaa tuen Firefox 6-versioon asti, joten uudempaa versiota ei kannata asentaa, vaikka selaimet tarjoavat päivityksiään melko usein aina uusimpaan versioonsa. /2, s. 14/

Firebug on helppo asentaa suoraan Firefox-selaimen kautta ja se integroituu selainikkunan työkaluksi. Käytettäessä Firebug sovellustestausta, konsoliin tulee testaustietoa ajetusta sovelluksesta melko paljon.



Kuva 11. Kehitysympäristön ja prosessien toiminta Vaadin-ohjelmistokehyksessä. /2, s. 12/

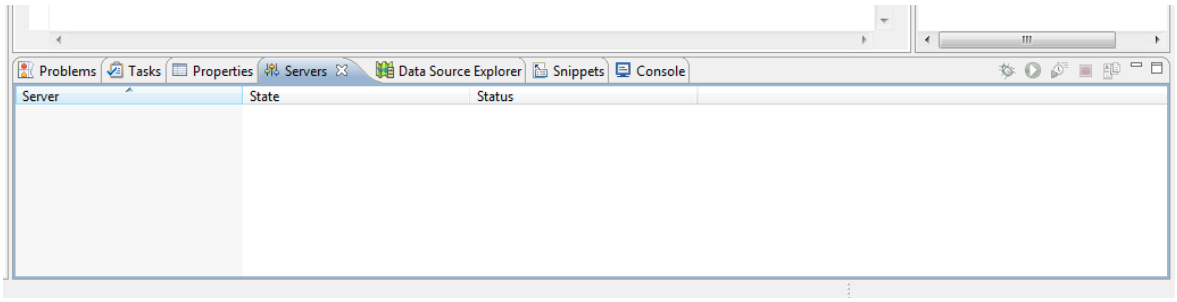
6.3. Projektin luominen ja testaaminen palvelimella

Tässä osiossa käydään läpi vaihe vaiheelta ohjekirjan mukaisen testiprojektin luonti ja testaus. Aloittelijan on mahdollista testata Vaadin-asennuspaketin sisältävät demo-ohjelmat sisäänrakennetulla palvelimella ilman erillistä Java-palvelimen asennusta. Ohjeet siihenkin liittyen on esitelty selkeästi, mutta tässä työssä ei enempää suoriteta demo-ohjelmien ajoa tai niiden vaiheistettua testausta (debuggausta).

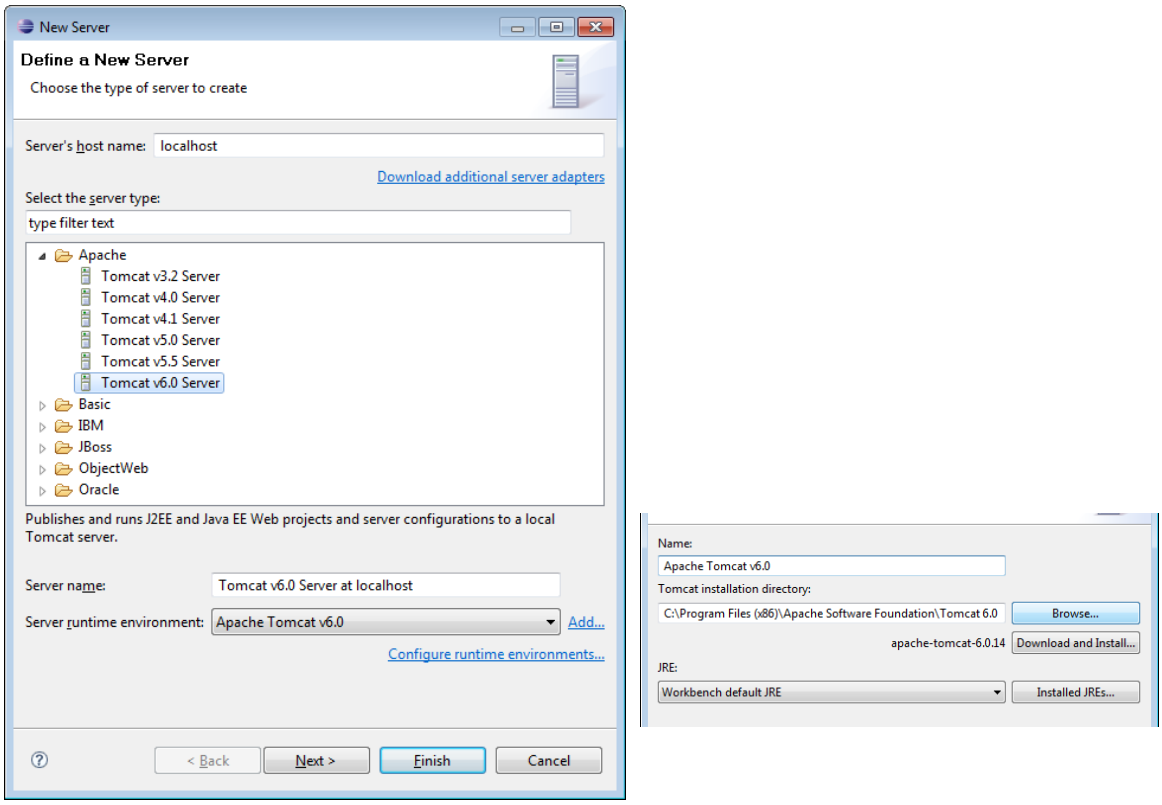
Ensimmäisen kerran avattaessa Eclipse IDE-ohjelmaa, ohjelma ehdottaa ja määrittelee työskentelytilan (workspace) eli kansion tehtäväksi omalle koneelle oletusarvoisesti. Uusia tilakansioita voi luoda projektikohtaisesti uusia niin monta kuin haluaa, tai käyttää vain yhtä, mihin sen sitten omalle koneelle haluaa asentaa. Oletusarvoinen paikka käy hyvin.

6.3.1. Tomcat-palvelimen käyttöönotto

Ennen uuden projektin luomista kannattaa ottaa asennettu palvelin käyttöön Eclipsen Java EE ï tilassa, ohjelman alapalkin ikkunasta. Tässä vaiheessa oletetaan jo, että kaikki muut toimenpiteet on tehty ohjekirjan mukaan ohjelmistokehyksen käyttöönotossa; asennukset, komponenttipäivitykset kehitysympäristön sisällä (3 Help 3 Software Updatesé) jne. Tätä määrittelyä ei ohjekirjan mukaan tehdä jo tässä vaiheessa, mutta tämä voi olla kuitenkin helpompi tapa, ennen kuin vasta sitten projektin luomisvaiheessa.

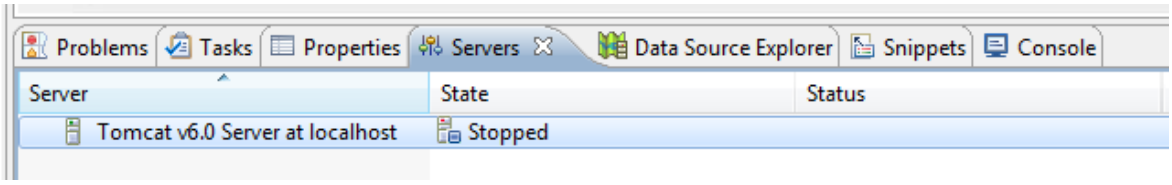


Palvelimen määrittely aloitetaan Servers-ikkunan päällä valinta hiiren oikealla painikkeella 3 New 3 Server.



New Server ikkunassa tarjotaan käyttöympäristöön sopivia Apachen Tomcat versioita, joista valinta tehdään. Add-linkistä avautuvaan ikkunaan määritellään polku omalle asennetulle Tomcat-palvelimelle. Se voidaan hakea selaamalla omaa järjestelmää Browse. Samasta ikkunasta voi ladata ja asentaa ohjelman tarjoaman Tomcat-version, ellei jostain syystä palvelinasennusta ole suoritettu ennakkoon. Ohjelman tarjoama versio ei ole välttämättä uusin päivitysversio suositellusta palvelinohjelmasta.

Ennen hyväksymistä ilmestyy vielä ikkuna, jossa palvelimelle ehdotetaan siirrettäväksi luotuja projekteja. Tässä vaiheessa kun niitä ei ole vielä luotu, niin vaiheen voi ohittaa kuittaamalla Finish.



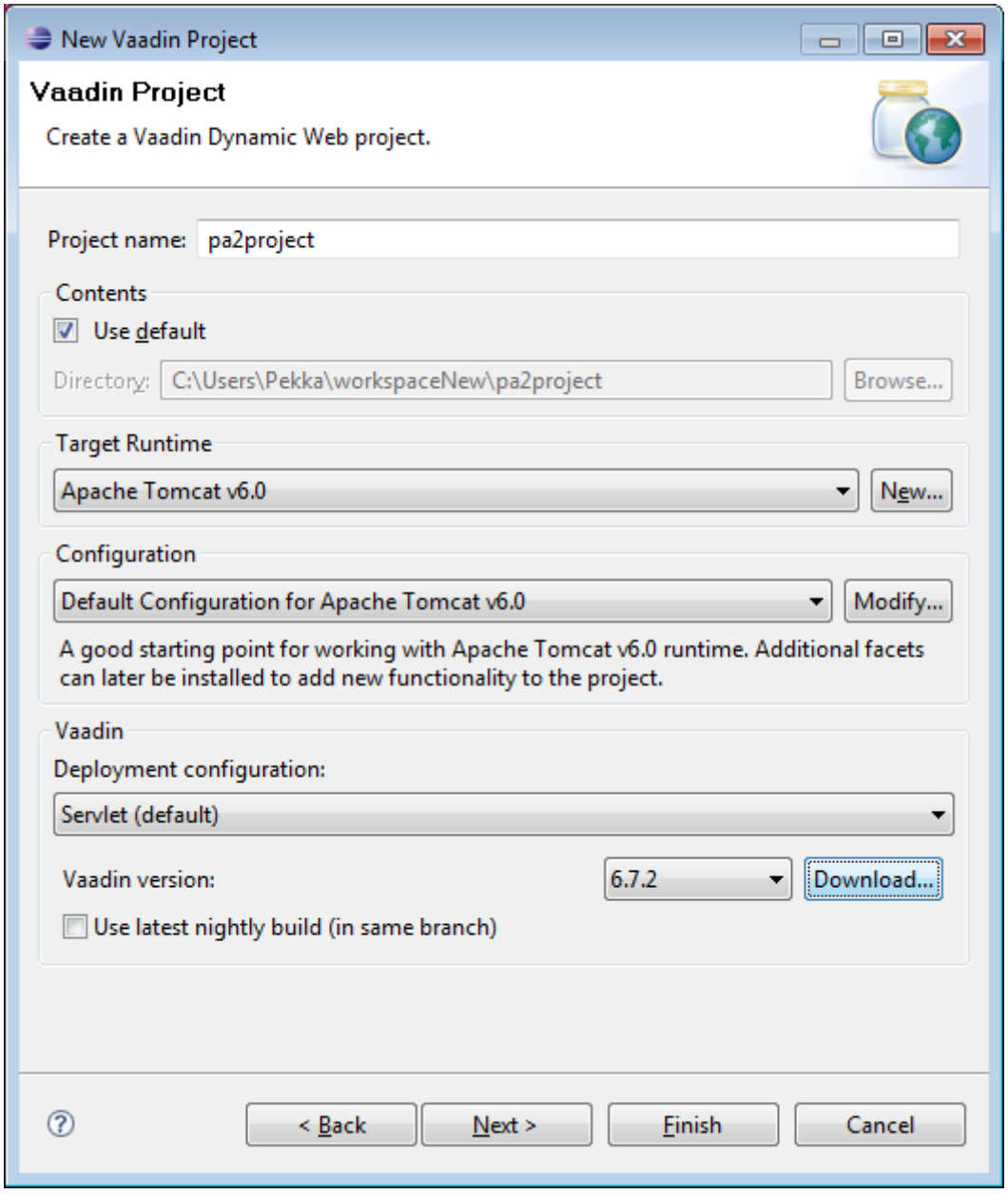
Kuva 12. Määritelty palvelin on Eclipsen alavalikon Servers-ikkunassa valmiina käyttöön.

6.3.2. Uuden projektin luominen

Vaadin-projekti luodaan monista ohjelmista varsin tutulla tavalla, 3 File 3 New 3 Project. Tämän jälkeen voi seurata ohjekirjaa tai kokeilla muita vaihtoehtoja, jos rahkeet riittävät. Projektin luominen on helppoa, mutta mitä sen jälkeen tehdään, on kiinni sovelluskehittäjän taidoista.

Vaadin-projektin luominen aloitetaan ja selataan ilmestyvästä ikkunasta Vaadin-kansion alta. Toinen tapa on 3 File 3 New 3 Other ja sen jälkeen Vaadin-kansion alta löytyykin enemmän vaihtoehtoja, Vaadin-projektin lisäksi.

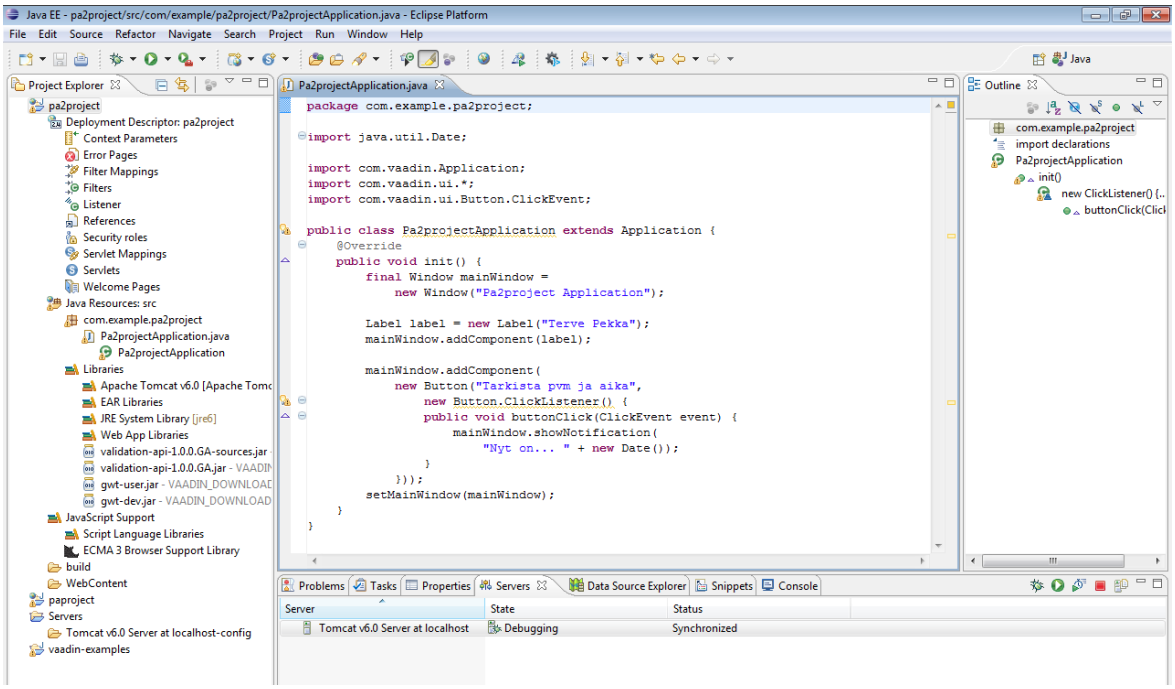
Kun edellä mainitut toimenpiteet Vaadin-projektin luomiseen on tehty, avautuu muutama määriteltävä ikkuna. Tässä on malliksi vain ensimmäinen avautuva ikkuna, joka on tärkein. Seuraavat ikkunat voi mennä kuittaamalla eteenpäin. Niissä määritellään hakemistotiedot ja ohjelmaan liittyvät luokat, jotka tulevat projektin nimen perusteella.



Ensin määritellään projektin nimi, jonka tulee olla sopiva nimi, kuten yleisesti ohjelmoinnissa suositellaan käytettäväksi. Tässä tapauksessa luodaan jo toinen testiprojekti uuteen tilakansioon. Palvelimen määrittely tulee nyt automaattisesti sen mukaan, mikä palvelin aiemmin määriteltiin ja otettiin käyttöön, sekä konfigurointi sen mukaan.

Vaadin-kirjaston versio on projektikohtainen ja tähän testiprojektiin on valittu uusin päivitetty versio. Lataamalla saa käyttöönsä asennettua minkä tahansa aiemman version, jolla projektia haluaa työstää.

Uuden projektin luomisen jälkeen Eclipsen graafinen käyttöliittymä näyttää alla olevan kuvan 13 mukaiselta. Vasemman puoleisessa valikossa näkyy avattuna kyseisen testiprojektin kansiot ja niiden sisältö. Listan lopussa näkyy aiemmin tehty projekti, määritelty palvelin ja vaadin-asennuspaketti (vaadin-examples), jossa on demo-ohjelmat ja paljon muita tarpeellisia esimerkkejä.



Kuva 13. Eclipsen integroidun kehitysympäristön graafinen käyttöliittymä.

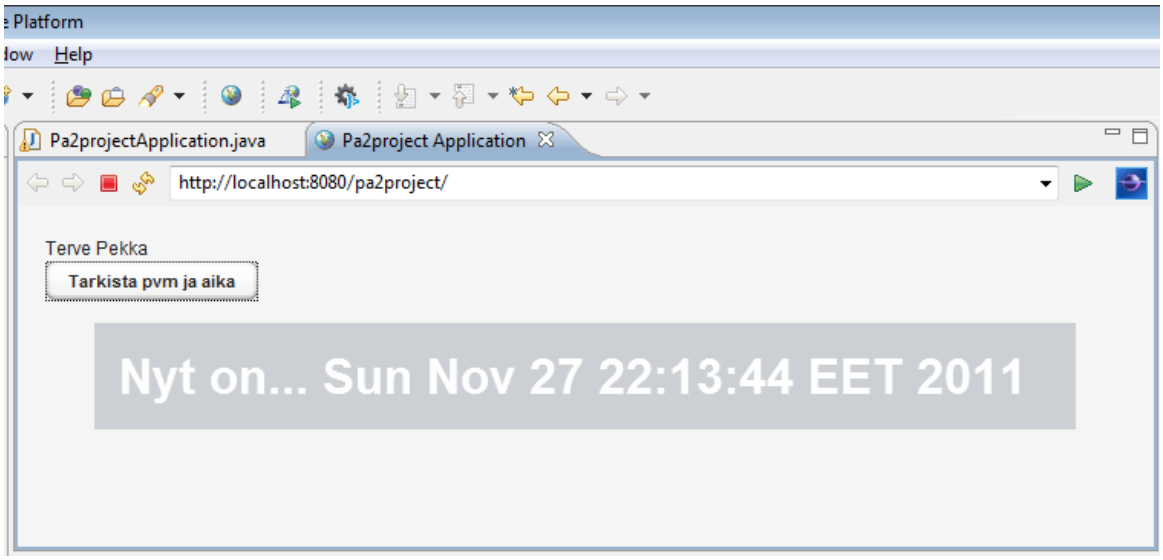
Pa2projectApplication.java sovelluksen aloituskoodiksi generoituu automaattisesti Hello Vaadin user ï tervehdysteksti-sovellus. Yksinkertaisessa testiprojektissa koodia on haluttu muokata ohjeiden mukaan hieman pitemmälle lisäämällä painike, joka näyttää päivämäärän ja ajan. Ohjekirjan koodi ei ole suoraan toimiva, vaan joitakin korjauksia on tehtävä, jotta virheilmoitukset saadaan pois. Toimivan testisovelluksen koodi voisi näyttää esimerkiksi kuten alla ja kuvan 13 keskimmäisessä ikkunassa:

```
package com.example.pa2project;

import java.util.Date;
import com.vaadin.Application;
import com.vaadin.ui.*;
import com.vaadin.ui.Button.ClickEvent;

public class Pa2projectApplication extends Application {
```

```
@Override
public void init() {
            final Window mainWindow =
                        new Window("Pa2project Application");

            Label label = new Label("Terve Pekka");
            mainWindow.addComponent(label);

            mainWindow.addComponent(
                        new Button("Tarkista pvm ja aika",
                                    new Button.ClickListener() {
                        public void buttonClick(ClickEvent event) {
                                    mainWindow.showNotification(
                                    "Nyt on... " + new Date());
                        }
                        }));
            setMainWindow(mainWindow);
            }
}
```

6.3.3. Testisovelluksen testaus

Sovelluksen ensimmäinen testaus palvelimella aloitetaan klikkaamalla projektin nimen päällä hiiren oikeanpuoleista painiketta. Tämän jälkeen valikosta valitaan 3 Debug as 3 Debug on Server, jonka jälkeen avautuvien ikkunoiden kautta haluttu projekti siirretään palvelin käyttöön vasemmasta laatikosta oikeanpuoleiseen laatikkoon Add-komennolla. Ja testataan sovellus kuittaamalla viimeinen ikkuna 3 Finish.



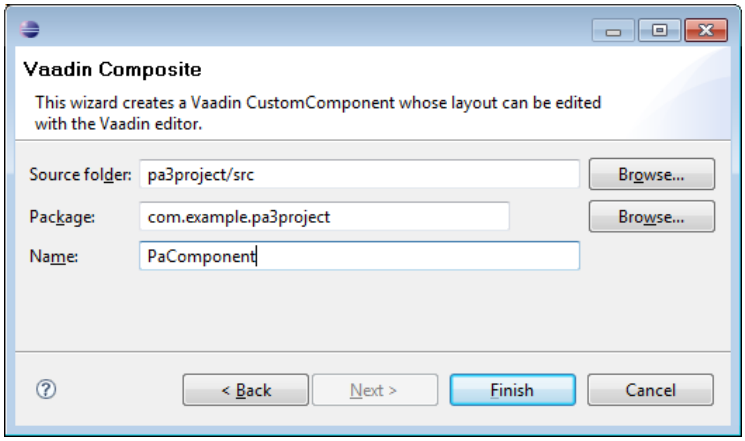
Kuva 14. Testauksen tulos näytetään suoraan Eclipsen sisällä, samassa ikkunassa kuin koodikin on. Klikkaamalla painiketta saadaan haluttu toiminto.

Sama tulos saavutetaan vastaavalla komennolla projektin päältä 3 Run as 3 Run on Server. Kuitenkin testauksessa on syytä varmistaa, ettei palvelin ole päällä koko ajan ja mahdollisesti muita ajoja on käynnissä eri asetuksilla. Aiemmin kuvatusta alavalikosta, jossa palvelinkin on, voidaan hallita palvelimen toimintoja siellä olevilla työkaluilla. Niiden avulla voi lopettaa (terminate) muut aiemmin testatut ajot ja poistaa konsolista (Console) niiden lokitiedot, sekä pysäyttää ja käynnistää palvelimen.

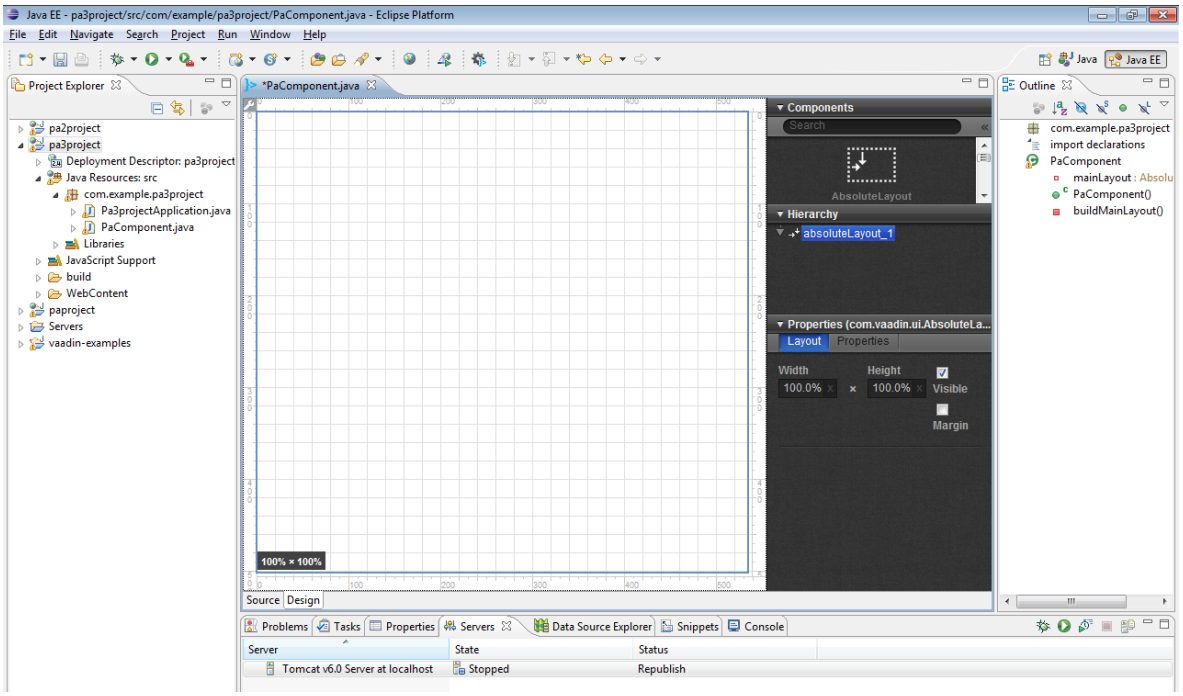
6.4. WYSIWYG-editorin käyttö

Vaadin on ottanut käyttöön myös visuaalisen tavan tehdä räätälöityjä sovelluksia tai pelkkiä komponentteja WYSIWYG-editorilla. Aluksi kokeellinen versio on kehittynyt toimivaksi kokonaisuudeksi 6.4-versiostaan alkaen. Editori generoi Java-koodia työskentelyn mukaan ja koodia voi tietysti muokata sekä täydentää tarpeen mukaan haluttujen toimintojen saavuttamiseksi. Tällainen työkalu lienee tuttu apuväline monissakin ohjelmistoissa, mm. avoimen lähdekoodin julkaisujärjestelmissä (CMS). Samalla tapaa toimivat monipuolisemmat graafiset web-sivujen suunnitteluohjelmat, joissa voi toimia sekä lähdekoodi- että suunnittelunäkymässä. /2, s. 182/

Sovelluksen luominen aloitetaan tuttuun tapaan 3 File 3 New 3 Other. Vaadin-kansiosta valitaan 3 Vaadin Composite. Seuraavassa alla olevassa ikkunassa määritellään, mihin projektiin luotava sovellus liittyy eli Vaadin-projekti on oltava luotuna valmiiksi. Tässä tapauksessa on luotu uusi testiprojekti.



Ikkunaan ilmestyy PaComponent.java-sovelluksen lähdekoodinäkymä. Ikkunan alaosasta voidaan vaihtaa näkymä WYSIWYG-editorille eli suunnittelunäkymään (Design).



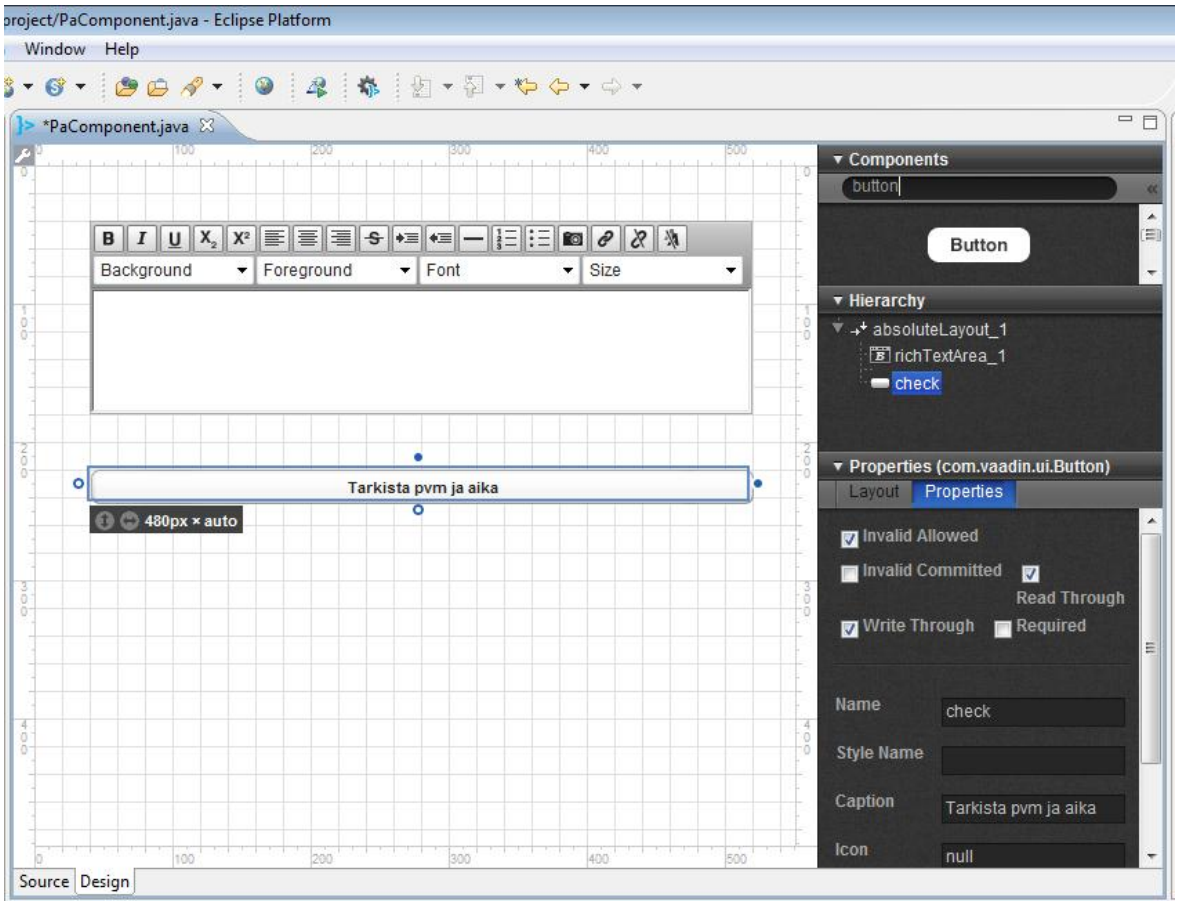
Kuva 15. WYSIWYG-editorin näkymä alun määrittelyjen jälkeen.

Suunnittelunäkymän käytettävät komponentit ovat kuvassa 15 tummassa valikossa oikealla. Komponentteja voi tehdä haulla, koska niitä on suhteellisen paljon. Seuraavana valikossa on Hierarkia-osio, jossa näytetään kaikki lisätyt komponentit. Alimmassa valikossa tehdään komponenteille ja koko pohjalle muokkaukset ja muut määrittelyt.

Seuraavaksi voidaan aloittaa räätälöidyn sovelluksen teko. Ajan puutteen vuoksi ja vain koemielessä rakennetaan parista komponentista sama yksinkertainen sovellus kuin alussa. Ohjekirjassa tätä ei ole tehty, vaan siinä on muulla tavalla perehdytty syvällisemmin komponenttien käyttöön ja muokkaukseen, joita ei tässä työssä tämän laajemmin käydä läpi.

Components-valikosta valitaan richTextArea- ja Button-komponentti ja raahataan ne pohjaan haluttuun paikkaan. Nimetään esimerkiksi painike halutulla tavalla Properties-valikossa, oletusarvoisesti se olisi koodissa button_1 ja painikkeessa Button. Muu

muokkaus onnistuu Layout-valikossa aktiivisen alueen tai komponentin osalta, joko syöttämällä valikossa arvoja tai tekemällä muutoksia manuaalisesti tarttumalla komponenttien kohdistuspisteisiin.



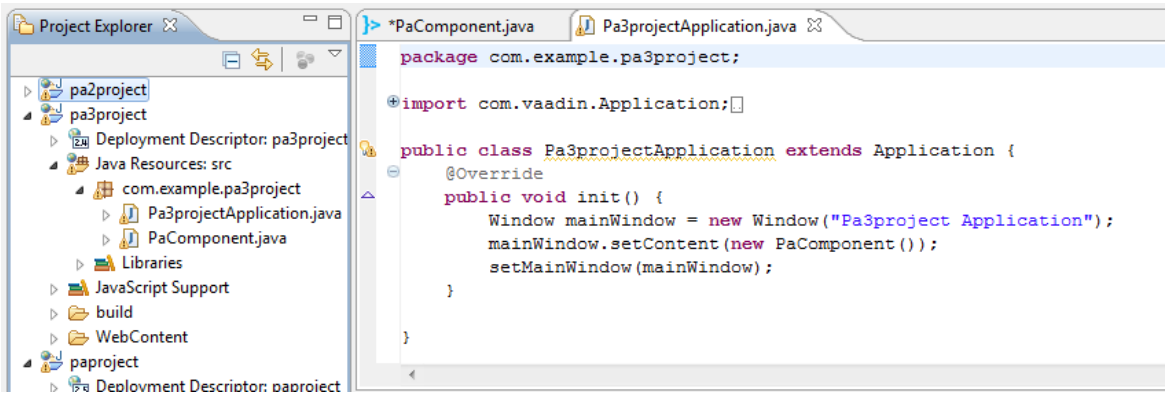
Kuva 16. Komponenttien sijoittelua ja määrittelyä WYSIWYG-editorilla.

Kuvan 16 mukaisen sijoittelun vastaava koodinäkymän voi katsastaa Source-valinnalla ikkunan alaosasta. Koodiin lisätään vielä painikkeelle toiminto ja kokonaisuudessaan generoitunut koodirivi näyttää ikkunassa seuraavanlaiselta:
(huom. tässä lähes suorana tekstinä, ilman oikeaoppista koodin rivitystä ja jaottelua)

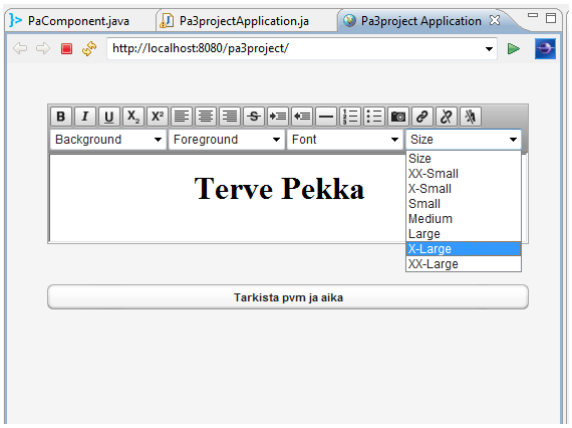
```
package com.example.pa3project;

import java.util.Date;

import com.vaadin.annotations.AutoGenerated;
import com.vaadin.ui.AbsoluteLayout;
import com.vaadin.ui.Button;
import com.vaadin.ui.CustomComponent;
import com.vaadin.ui.RichTextArea;
```

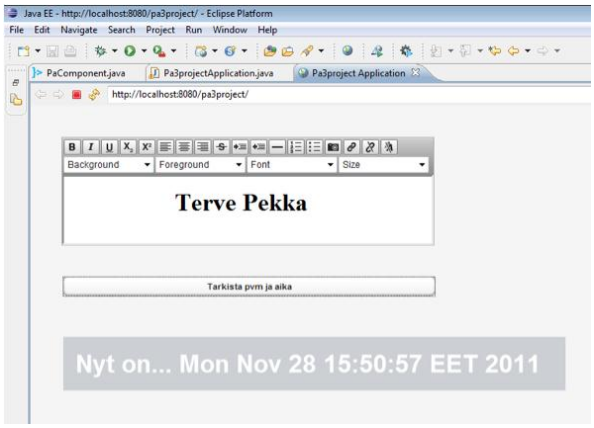
```
import com.vaadin.ui.Button.ClickEvent;

public class PaComponent extends CustomComponent {

	@AutoGenerated
	private AbsoluteLayout mainLayout;
	@AutoGenerated
	private Button check;
	@AutoGenerated
	private RichTextArea richTextArea_1;

	public PaComponent() {
	buildMainLayout();
	setCompositionRoot(mainLayout);

	check.addListener(new Button.ClickListener(){

			public void buttonClick(ClickEvent event) {
				getWindow().showNotification(
					"Nyt on... " + new Date());
			}
			});
	}

	@AutoGenerated
	private AbsoluteLayout buildMainLayout() {
			// common part: create layout
			mainLayout = new AbsoluteLayout();
			mainLayout.setImmediate(false);
			mainLayout.setWidth("100%");
			mainLayout.setHeight("100%");
			mainLayout.setMargin(false);

			// top-level component properties
			setWidth("100.0%");
			setHeight("100.0%");

			// richTextArea_1
			richTextArea_1 = new RichTextArea();
			richTextArea_1.setImmediate(false);
			richTextArea_1.setWidth("480px");
			richTextArea_1.setHeight("140px");
mainLayout.addComponent(richTextArea_1, "top:40.0px;left:40.0px;");

			// check
			check = new Button();
			check.setCaption("Tarkista pvm ja aika");
			check.setImmediate(true);
			check.setWidth("480px");
			check.setHeight("-1px");
mainLayout.addComponent(check, "top:220.0px;left:40.0px;");

			return mainLayout;
	}

}
```

Jotta testauksen pystyy suorittamaan palvelimella, tulee aiemmin luodun pa3-projektin ottaa käyttöön luotu PaComponent. Käytännössä se tarkoittaa koodin lisäämistä Pa3projectApplication.java ï sovellukseen, alla esitetyllä tavalla.



Tämän jälkeen voidaan sovellusta testata tai ajaa palvelimella aiemmin kuvatuilla tavoilla, joko òDebug on Serverò tai òRun on Serverò.





Ylläolevissa kuvissa näkyy pikaisen testin lopputulos. Molemmissa on testattu komponenttien toimivuutta. Oikeanpuoleisessa kuvassa on otettu käyttöön kokosivun näkymä, joka onnistuu klikkaamalla sovelluksen nimeä yläpalkista. Muussa tapauksessa päivän ja ajan näyttö olisi mennyt painikkeen päälle.

Editori tuntui toimivalta ja myös helpolta käyttää näin aloittelijan näkökulmasta ajateltuna, joskin tässäkin on sekoitettu termeillä hieman lisää komponentin (CustomComponent)

luomista. Vaadin käyttää tällaisen sovelluksen luomiseen Composite-termiä, joka kuvaa ehkä paremmin komponenttien yhdistelmää.

7. TULEVAISUUDEN NÄKYMIÄ

Vaadin ei ole vielä kovinkaan tunnettu ja maailmanlaajuisesti kovin yleisesti käytössä oleva ohjelmistokehys. Kehityssuunta on kuitenkin kokoajan eteenpäin ja uusia ominaisuuksia sekä integroituja sovelluksia on otettu käyttöön. Viimeisimpänä Vaadin liitännäinen Spring Roo-kehitystyökaluille. Spring Roo on yksi monesta SpringSource-yhteisön projekteista. Yhteisön taustalla häärii VMWare, jolla on laajasti osaamista ja muitakin ohjelmistotuotteita erilaisiin ympäristöihin.

Kilpailu on kovaa ja yhä enemmän ohjelmistot yhdistyvät tavalla tai toisella. Google App Enginen ja Google Web Toolkitin käyttö osana omaa ohjelmistoa on tästä hyvä esimerkki. Google on myös kehittänyt oman Go-ohjelmointikielen, joka on julkaistu avoimella BSD-lisenssillä. Mitä se sitten tuo tullessaan, jää nähtäväksi.

Matt Raible on luetellut ohjelmistokehyksille esityksessään jotain merkityksellisiä teknologioita ja niiden vaikutuksia, joita kehittäjien tulisi ottaa huomioon tulevaisuudessa.

- Älypuhelimet ja tabletit sekä niiden käyttöjärjestelmien tuki.
- HTML5 ja sen tuomat kehitysnäkymät.
- Web-sovellukset työpöydällä, kuten Adobe Air.
- Tuki liitännäisille, unohtamatta selaimia.
- Ja lopuksi tärkein eli API ja siihen liittyen nopeus. /8/

Yllä olevaan supistettuun listaan voisi lisätä vielä Pilvipalvelut, johon varsinkin yritysohjelmistoratkaisut ovat pikku hiljaa siirtymässä. Samalla tavalla myös muut voivat käyttää niitä ja yhä useammat ohjelmistopalveluyritykset tarjoavat niitä.

Vaadin on ottanut jo huomioon edellä lueteltuja asioita. Lokakuussa 2011 se julkaisi TouchKit 2.0, joka tukee iOS (iPad), Android ja monia muita uusia komponentteja. Samaan aikaan julkaistiin päivitys Vaadin 6.7 Geneve, joka yhdisti lisäosia (Add-ons) kuten TreeTable, SQLContainer, Chameleon ja lisäsi samalla muita pienempiä ominaisuuksia. Lisää kehityssuunnitelmia vuoteen 2012 asti on nähtävissä osoitteessa: https://vaadin.com/roadmap.

8. YHTEENVETO

Vaadin oli mielenkiintoinen kokemus, vaikka sen tutkimus- ja testaustyö veivät aikaa yllättävän paljon. Se johtui siitä, ettei minulla ollut aiempaa kokemusta tällaisista Java-pohjaisista ohjelmistokehyksistä. Aiemmin olen käyttänyt web-sivustojen tekemiseen jonkin verran julkaisujärjestelmiä (Joomla ja WordPress), sekä muita yksittäisiä graafisia web-ohjelmia kuten Adoben Dreamweaver- ja Flash-ohjelmaa. Se kokemus auttoi jonkin verran, samoin kuin opintoihini kuuluva olio-ohjelmoinnin peruskurssi.

Englanninkielinen Vaadin-ohjekirja oli selkeästi kirjoitettu ja sisälsi suhteellisen helposti ymmärrettävää tekstiä. Erikoissanoja ja erikoisia sanontoja oli kuitenkin paljon, johtuen aiheen teknillisestä puolesta. Tätä kirjoittaessa, ohjekirjasta on julkaistu päivitetty versio, Book of Vaadin, 4th Edition, ohjelmistoversiolle 6.6. Tätä opinnäytetyötä aloittaessa käytössä oli 6.4-version tulostettu ohjekirja, jota on käytetty tämän opinnäytetyön pääasiallisena lähteenä loppuun asti. Poikkeavuuksia saattaa siis olla uusiin kirjan versiopäivityksiin ja ohjelmiston ominaisuuksiin. Myös muut tutkitut lähteet olivat pääosin englanninkielisiä, jos niiden avulla halusi syventää tietämystä juuri tällaisista Java-pohjaisista ohjelmistokehitysratkaisuista.

Tämän tyyppisen teknologian termistön kirjo on laaja ja moniulotteinen. Sovelluksiin ja sivustoihin liitettävien lisäosien nimeäminen on varmasti paljolti kiinni tarkoitusperästä ja siitä, mihin kategoriaan ne on haluttu sijoittaa valmistajan toimesta ja minkälaisia toimintoja niillä on. Silti toiminnoiltaan samanlaisia lisäosia kutsutaan eri ohjelmistoissa eri termein. Joissakin julkaisujärjestelmissä lisäosat jaotellaan moduuleihin, komponentteihin ja liitännäisiin. Joissakin muissa julkaisujärjestelmissä vastaavasti samoja òohjelmapalikoitaò kutsutaan ja nimetään widgeteiksi. Valmista pohjaa kutsutaan ohjelmistosta riippuen teemaksi tai templateksi, ja joissakin tapauksissa myös layoutiksi, joka ei välttämättä tarkoita valmiiksi luotua pohjaa sinänsä.

Vaadin toi uuden näkökulman tähänkin termistösoppaan. Siinä käytetään komponentista kolmea eri tapaa ilmaista periaatteessa sama sovelluksen lisäosa. Palvelinpuoli käyttää komponentteja niiden tekemiseen, muokkaamiseen ja ajamiseen. Asiakaspuoli toimii

GWT:n myötävaikutuksella widgeteinä. Oma räätälöity komponentti eli CustomComponent työstetään Vaadin-valikosta termillä Vaadin Composite.

Aloittelijalla voi siis termit mennä helposti sekaisin, vaikka samasta asiasta on kyse monessakin tapauksessa. Joka tapauksessa helpoin tapa on tehdä sovelluksia WYSIWYG-editorilla siinä tapauksessa, jos laajempaa ohjelmointi osaamista ei ole. Valmiita koodeja ja muita komponentteja on helposti kopioitavissa Vaadin-paketista. Muita lisäosia voi ladata ja ottaa käyttöön Vaadin-kotisivulta. Työssä tehtyjen yksinkertaisten testien osalta ei ollut kovinkaan paljon ongelmia. Yksi mainittava tuli eteen testattaessa sovelluksen tekemistä WYSIWYG-editorilla. Suunnittelupohjan käyttöönotossa Eclipse-ohjelma herjasi Mozillan XULRunner-ajoympäristön versiota ja se olisi pitänyt asentaa uudemmaksi, jos haluaisi editorin täyden toimivuuden. Pitkään etsimisen jälkeen ei löytynyt suoraan ratkaisua, vaan yleisesti oli tiedossa, ettei Mozillan Firefox-selaimen liitännäiseen ollutkaan vielä löytynyt ratkaisua Mozillan kehittäjiltä, eikä selityksiä löytynyt edes Eclipsen omilta sivuilta. Linkkejä löytyi tukuttain, mutta turhaan. Tästä huolimatta Vaadin suosittelee selaimeksi Firefoxia.

Herjasta huolimatta, editori ja sovelluksen ajo toimi moitteettomasti. Kyse oli ilmeisesti 32/64-bittisten ohjelmien käytöstä ja niihin liittyvistä yhteensopivuusongelmista. Järkevin tapa on ainakin vielä suosia 32-bittisten ohjelmien tai sovellusten käyttöä, varsinkin kun kyseessä on ohjelmistokehys ja kehitysympäristö, jossa monet tekijät toimivat keskenään.

Vaadin sopii hyvin varsin kattavaan ja paisuvaan Java-pohjaisten ohjelmistokehysten joukkoon. Vaadin on hyvä monenkin mittarin perusteella, mutta kuuluu vielä pienten joukkoon suosion perusteella. Tutkimusten ja vertailujen perusteella Vaadin on epäilemättä varteenotettava tekijä puhuttaessa Java-pohjaisista web-ohjelmistokehyksistä. Mihin asti rahkeet riittävät tai kuinka korkealle ohjelmistokehittäjät sen arvostavat tulevaisuudessa, riippuu täysin omasta sitoutuneesta ja aktiivisesta kehittäjäyhteisöstä. Eväät ovat olemassa, samoin tietotaito.

Vaadin on myös naarasporo, josta logoon on otettu mukaan aaltosulkeet kuvaamaan poronsarvia, käännettynä. Suomalaisuus on vahvasti esillä. Siitäkin voi olla ylpeä.

9. LÄHDELUETTELO

/1/ Asiakas/palvelin-arkkitehtuuri, Luentomateriaali, Teknillinen korkeakoulu, [PDF-dokumentti], [http://www.cs.hut.fi/Opinnot/T-106.270/2002/Luennot/resources/client_server.pdf] 15.9.2011.

/2/ Grönroos, Marko, Book of Vaadin: Vaadin 6.4, [PDF- tai HTML-dokumentti], [https://vaadin.com/book] 4.1.2011.

- Uusi versio julkaistu, Book of Vaadin: 4th Edition (Vaadin 6.6).

/3/ Java Advantages and Disadvantages, Java Home/Web-dot-dev ï yhteisöblogit, [WWW-dokumentti], [http://www.webdotdev.com/nvd/content/view/1042/204/] ja [http://java-work.blogspot.com/2008/08/java-advantages-and-disadvantages.html] 20.7.2011.

/4/ JVM Web Framework Matrix (+Weighted), Google Docs, [WWW-dokumentti], [http://bit.ly/jvm-frameworks-matrix] 16.7.2011.

/5/ Lehtinen, Joonas, Vaadin Scalability, [HTML-esitys], [http://www.slideshare.net/codento/vaadin-scalabilityslides] 10.8.2011.

/6/ Lyytikäinen, Susanna, Tietoviikko: Java menettää suosiotaan ohjelmointikielenä, [WWW-dokumentti], [http://www.tietoviikko.fi/kaikki_uutiset/java+menettaa+suosiotaan+ohjelmointikielena/a701996] 11.10.2011.

/7/ Nikulainen, Kalevi, IT-uutiset: IT Millistä kehkeytyi Vaadin, [WWW-dokumentti], [http://www.itnyt.fi/it-uutiset/2556-nbspit-millistauml-kehkeytyi-vaadin] 6.8.2011.

/8/ Raible, Matt, Presentations: Comparing JVM Web Frameworks (Rich Web Experience 2010), [PDF-dokumentti tai HTML-esitys], [http://www.slideshare.net/mraible/presentations] 16.7.2011.

/9/ Raible, Matt, Presentations: Comparing JVM Web Frameworks (TSSJS 2011), [PDF dokumentti tai HTML-esitys], [http://www.slideshare.net/mraible/presentations] 20.11.2011.

/10/ Turto, Tuomas, Tampereen teknillinen yliopisto, Luentomateriaali, Web-ohjelmointi: Web-ohjelmointikehykset ja sivupohjamoottorit, [PDF-dokumentti], [http://www.cs.tut.fi/~seitti/media/pdfs/luento-3.pdf] 15.9.2011.

/11/ Vaadin, Comparison [WWW-dokumentti], [https://vaadin.com/comparison]14.7.2011.

/12/ Wikipedia, Java (programming language), [WWW-dokumentti], [http://en.wikipedia.org/wiki/Java_(programming_language)] 23.8.2011.

/13/ Wikipedia, Comparison of Web application frameworks, [WWW-dokumentti], [http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks] 2.7.2011.

/14/ ZeroTurnaround, Java EE Productivity Report 2011, [WWW-dokumentti], [http://zeroturnaround.com/java-ee-productivity-report-2011/] 5.11.2011.

10. LIITELUETTELO

[edit]

Java

Project	Language	Ajax	MVC framework	MVC Push/Pull	i18n & l10n?	ORM	Testing framework(s)	DB migration framework(s)	Security Framework(s)	Template Framework(s)	Caching Framework(s)	Form Validation Framework(s)
Spring	Java	Yes	Yes	Push	Yes	Hibernate, iBatis, etc	Yes, mock objects & unit tests		Spring Security (formerly Acegi)	JSP, Commons Tiles, Velocity, Thymeleaf, etc.	ehcache etc.	Commons Validator
Apache Click	Java	JQuery	Page Oriented	Pull	Yes	Hibernate and Cayenne	Yes		pluggable	Velocity and JSP	Cached Templates	Built-in validation
Apache Sling	Java	Yes	Yes	Push & Pull		Uses JCR content repository			Yes	Yes	Yes	
Apache Struts	Java	Yes	Yes	Push & Pull	Yes	Yes	Unit Tests			Yes		Yes
Apache Wicket	Java	Yes, extensions for YUI, ExtJS and more	Modular event driven	Pull	Yes	Yes, with extensions	Yes, mock objects, unit and integration tests through an extension		Yes	Yes	Yes	Yes
Aranea	Java	Yes		Pull	Yes	Yes				mapping-applications using contributions from users to advantage		
FormEngine	Java	Yes			Yes	own connector API						AJAX validation on server and form state update
ItsNat	Java	Yes	event driven	Push	using Java i18n		external and built-in		pluggable	pure HTML/SVG	page caching	normal Java
JavaServer Faces	Java	Yes	Yes	Pull	Yes	Yes, with extensions	JUnit		Yes	Facelets, JSP	Yes	Native validators and integration with Bean Validation
JBoss Seam	Java	Yes	Yes	Pull	Yes	JPA, Hibernate	JUnit, TestNG		JAAS integration, Drools, Hibernate Filters, OpenID, CAPTCHA	Facelets	JBoss Cache, EHcache	Hibernate Validator
Jspx-bay	Java	Yes	Page oriented			Own API			JAAS integration	Master/Content Pages		Yes, Internal UI validation controls
JVx WebUI	Java	Yes	Model Driven		Yes	Yes, pluggable	JUnit		Yes	Single sourcing		Yes, pluggable
OpenXava	Java	Yes	Model Driven		Yes	JPA, Hibernate and EJB2 CMP	JUnit	Hibernate tools	uses JSR-168 portal security	UI is automatically generated	uses portal and JPA caching	Yes

>>

Project	Language	Ajax	MVC framework	MVC Push/Pull	ORM	Testing framework(s)	DB migration framework(s)	Security Framework(s)	Template Framework(s)	Caching Framework(s)	Form Validation Framework(s)
OpenXava	Java	Yes	Model Driven		Yes	JUnit	Hibernate tools	uses JSR-168 portal security	UI is automatically generated	uses portal and JPA caching	Yes
Play	Java	Yes	Yes	Push and Pull	Yes	JUnit, Selenium (Software)	Yes	via Core Security module	Yes	Yes	Server-side validation
RIFE	Java	DWR	Yes	Push & Pull	Yes	Out of container testing		Yes	Yes	Integration with Terracotta	Yes
Stripes	Java	Yes	Yes	Pull	Yes	JPA, Hibernate	Yes	framework extension	Yes		Yes
Apache Tapestry	Java	Yes	Yes	Pull	integrated with Hibernate (tapestry-hibernate module)	Yes		tapestry5-acegi library	Yes		built-in validation system
Vaadin	Java	GWT		Pull	Yes	Yes			Yes		Yes
Wavemaker	JavaScript (client), Java (server)	Dojo Toolkit	Yes	Push	Hibernate (Java)	Dojo Toolkit	JUnit	Hibernate (Java)	Spring Security, Acegi, Role-based access control	Dojo Toolkit	Dojo Toolkit	Regular expression, schema-driven validation
WebObjects	Java	Yes	Yes	Push & Pull	Yes	EOF	WOUnit (JUnit), TestNG, Selenium	in Project WONDER		Yes	Yes	Yes
ztemplates	Java JDK 1.5 or newer	integrates YUI, Google etc. with annotations	Yes	Push, multiple actions per URL	standard Java	use any J2EE ORM framework	Unit Tests		annotation based	Velocity, FreeMarker, JSP, others pluggable		AJAX validation on server and form state update (YUI, JSON)
Google Web Toolkit	Java, JavaScript	Yes			Yes	JPA with RequestFactory	JUnit (too early), jsUnit (too difficult), Selenium (best)	via Java	Yes			
ZK	Java, ZUML	JQuery	Yes	Push & Pull	Yes	any J2EE ORM framework	JUnitZTL	HibernateUtil & SpringUtil	Spring Security	Macro components & Composition	Yes	client and server

FEATURE	VAADIN 6.7	FLEX 4.5.1	GWT 2.4.0	JAVAFX 2.0	JQUERY 1.6.4	RICHFACES 4.0.0	SWING JAVA 7	WICKET 1.5.0	ZK 5.0.8
Core UI widgets	***	***	**	**	**'	**	**	*	***
No browser plug-in required	●		●		●	●		●	●
No JavaScript programming required	●		●	●		●	●	●	●
Framework extensions are done in Java	●		●	●			●		●
No HTML authoring required	●	●	●	●			●		●
No XML configuration required	●		●	●	●		●	●	●
UI programming on client-side	Java, Javascript	ActionScript	Java, Javascript	Java, JVM langua	Javascript	Javascript	Java, JVM languages	Javascript	Javascript
UI programming on server-side	Java, JVM languages	-	-	-	-	Java, JVM languages	-	Java, JVM languages	Java, JVM lan
Page request oriented					●	●		●	
Application oriented	●	●	●	●			●		●
Backed up by a corporation	●	●	●	●		●	●		●
Commercial support & guarantees available	●	●		●		●			●
Vendor offers experts services to complement	●					●			●
Free for commercial use	●	◔ '	●	●	●	●	●	●	● '
License	Apache 2.0	MPL 1.1	Apache 2.0	?'	MIT/GPL v2	LGPL	-	Apache 2.0	LGPL/Comme
Portlet support	●	●				◔ '		●	●
Google App Engine support	●	●	●		●	●	●	●	
Requires a Java Servlet container	●					●		●	●
Inherent search engine support						●		●	
Documentation	***	***	**	**	**	***	***	**	***
Eclipse IDE support	***	***'	***	*	*	***	***	**	***
NetBeans IDE support	**	*	**	***	*	**	***	**	**
IntelliJ IDEA support	*	*	**	*	*	**	***	**	*
Simple workflow with plain text editor					●				
Internet Explorer 6	●	◔ '	●		●	●	●	●	●
Internet Explorer 7-9	●	◔ '	●	◔ '	●	●	●	●	●
Mozilla Firefox	●	◔ '	●	◔ '	●	●	●	●	●
Google Chrome	●	◔ '	●	◔ '	●	●	●	●	●
Safari	●	◔ '	●		●	●	●	●	●
Opera	●	◔ '	●		●	●	●	●	●
Android	●	◔ '	●		●	●		●	●
iOS	●	◔ '	●		●	●		●	●
Touch event support	●	●	●		●				●
Specialized UI widgets for mobile	◔ '				◔ '				
Number of downloadable add-ons	215'	436'	-	-	6470'	-	-	54'	~20'
Data binding on server-side data	● '					●		●	●
Support forum activity (msg/month)	1375'	?	1507'	657'	5063'	843'	?	928'	857'